0

CDRL: B011
25March 1994

UNISYS

Acquisition Handbook - Update
Comprehensive Approach to Reusable Defense Software (CARDS)

Informal Technical Report

CARDS
Comprehensive Approach to Reusable Defense Software

STARS-VC-B011/001/00
25 March 1994

19918

CDRL: B011
25 March 1994

INFORMAL TECHNICAL REPORT For The SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

Acquisition Handbook - Update
Central Archive for Reusable Defense Software
(CARDS)

STARS-VC-B011/001/00
25 March 1994

Data Type: Informal Technical Data
Contract NO. F19628-93-C-0130
Line Item 0002AB

Prepared for:

Electronic Systems Center
Air Force Material Command, USAF
Hanscom AFB, MA 01731-2816

Prepared by:

DSD Laboratories, Inc.
under contract to
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

CDRL: B011
25 March 1994

INFORMAL TECHNICAL REPORT
For The
SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

Acquisition Handbook - Update
Central Archive for Reusable Defense Software
(CARDS)

STARS-VC-B011/001/00
25 March 1994

Data Type: Informal Technical Data

Contract NO. F19628-93-C-0130
Line Item 0002AB

Prepared for:

Electronic Systems Center
Air Force Material Command, USAF
Hanscom AFB, MA 01731-2816

Prepared by:

DSD Laboratories, Inc.
under contract to
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

CDRL: B011
25 March 1994

Data Reference: STARS-VC-B011/001/00
INFORMAL TECHNICAL REPORT
Acquisition Handbook - Update
Central Archive for Reusable Defense Software
(CARDS)

CDRL: B011
25 March 1994

Data Reference: STARS-VC-B011/001/00
INFORMAL TECHNICAL REPORT
Acquisition Handbook - Update
Central Archive for Reusable Defense Software
(CARDS)

Principal Author(s):

Leandro V. Delgado	*Date*
Robert J. Bowes	*Date*
Theresa R. Huber	*Date*
Robert O. Saisi	*Date*

Approvals:

System Architect: *Kurt Wallnau*	*Date*
Program Manager: *Lorraine Martin*	*Date*

(Signatures on File)

Data Reference: STARS-VC-B011/001/00
INFORMAL TECHNICAL REPORT
Acquisition Handbook - Update
Central Archive for Reusable Defense Software
(CARDS)

ABSTRACT

The Central Archive for Reusable Defense Software (CARDS) Program is a concerted Department of Defense (DoD) effort to transition advances in the techniques and technology of library-assisted, domain-specific software reuse into mainstream DoD software procurements.

There are four key elements to the CARDS approach:

1. Develop a CARDS knowledge-base for development of domain-specific reuse processes designed to support reuse based system development

2. Define a plan to perform technology transfer to other government organizations (Franchise Plan)

3. Implement the plan for technology transfer through franchising

4. Improve the knowledge-base and blueprint through continuous process improvement

The Franchise Plan will provide a description of reuse processes and instructions for tailoring development processes to implement domain-specific reuse. In addition, it will describe in precise steps a scenario for implementing a domain-specific library. Along with the Franchise Plan, organizations will be provided with three sets of documents: Reuse Adoption Handbooks, CARDS library operation and maintenance related documents, and training and education material. The Reuse Adoption Handbooks consist of: Direction Level Handbook, Acquisition Handbook, Engineer's Handbook, and the Component Developer's and Tool Vendor's Handbook. The Engineer's Handbook and the Component Developer's and Tool Vendor's Handbook are technically oriented; the Direction Level and Acquisition Handbooks emphasize the business aspects of reuse, including but not limited to strategy, schedule, risk, cost and rights issues. The CARDS Library Model Contracts/Agreements document provides a background of related legal issues, guidance to the CARDS Library Staff in applying the agreements to suppliers, subscribers and a starting point to develop library interoperability. The following paragraphs provide more of an overview to the Direction Level and Acquisition Handbooks.

The Acquisition Handbook is aimed towards all Government Program Managers and their support personnel, such as Contracting Officers and Administrators, procurement attorneys, and program control, involved in systems, subsystems and component acquisition and maintenance. The concepts discussed assume that the reader has at least three years experience in acquisition. This guidebook will assist them in incorporating software reuse into all phases of the acquisition life cycle, from concept exploration to Post Deployment Software Support (PDSS). It is not a

"cookbook" for every possible reuse issue or strategy, rather it is meant to help you develop and tailor reuse programs. The goal of the Acquisition Handbook is to encourage software reuse during the acquisition and maintenance portions of the lifecycle process, ranging from planning the acquisition strategy through awarding the contract to man aging the effort and follow-on support. Software reuse guidance will be presented by providing methods, examples, recommendations and techniques to implement various reuse strategies throughout the acquisition life cycle. The implications and affects of software reuse on the technical, management, cost, schedule, and risk aspects of a program/system during the acquisition process will be the foundation of this document

The Direction Level Handbook is directed towards acquisition executives of all the services to facilitate the institutionalization of software reuse. The audience of Program Executive Officers (PEOs), Designated Acquisition Commanders (DACs), and their supporting staff, are provided with a framework to assist them in establishing plans to manage reuse across their systems and to reach the goals outlined in the DoD Software Reuse Vision and Strategy document. Considerations are provided to assist in incorporating software reuse into the initial planning stages of an acquisition, as well as at critical points within the acquisition life cycle. The options provided the executive will allow him to gain the greatest benefits from software reuse while optimizing the use of shrinking resources.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **Department of the Air Force** **ESC/ENS** **Hanscom AFB, MA 01731-2816**	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT **DISTRIBUTION "A"**	**12 b. DISTRIBUTION CODE**

13. ABSTRACT (Maximum 200 words)

The Acquisition Handbook is to aimed towards all Government Program Managers and their support personnel, such as Contracting Officers and Administrators, procurement attorneys, and program control, involved in systems, subsystems and component acquisition and maintenance. The concepts discussed assume that the reader has at least three years experience in acquisition. This guidebook will assist them in incorporating software reuse into all phases of the acquisition life cycle, from concept exploration to Post Deployment Software Support (PDSS). It is not a "cookbook" for every possible reuse issue or strategy, rather it is meant to help you develop and tailor reuse programs. The goal of the Acquisition Handbook is to encourage software reuse during the acquisition and maintenance portions of the life cycle process, ranging from planning the acquisition strategy through awarding the contract to managing the effort and follow-on support. Software reuse guidance will be presented by providing methods, examples, recommendations and techniques to implement various reuse strategies throughout the acquisition life cycle. The implications and affects of software reuse on the technical, management, cost, schedule, and risk aspects of a program/system during the acquisition process will be the foundation of this document.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Software reuse is a process in which software resources are applied to more than one system. It can occur within a system (e.g., F-14A, B, ...), across similar systems (e.g., M1, Bradley, ...) or in widely different systems. [1]. All resources or components resulting from the various stages of the software development process have the capability of being reused. These components include: domain models, software architectures, product designs, and implementation components (source code, test plans, procedures and results, and system/software documentation).

This handbook provides a business framework for incorporating domain-specific software reuse into the acquisition life cycle for systems, subsystems and components. Information discussed can apply to all phases of the acquisition life cycle: from concept exploration to Post Deployment Software Support (PDSS). Guidance is provided to encourage software reuse, beginning with the establishment of an Acquisition Strategy Panel, through development of an Acquisition Plan to writing the contract and managing the effort, as well as follow-on support. This guidance is presented by providing recommendations, techniques and methods to implement various reuse strategies throughout the acquisition life cycle. The implications and effects of software reuse on the technical, management, cost, schedule, and risk aspects of a program/system during the acquisition and contractual processes are the foundation of this document.

All functional roles involved in system, subsystem and component acquisition are addressed. These functions include Program Managers, Contracting Officers and Administrators, legal support and program control personnel and other relevant program management personnel. Program Managers are presented with various information to assist them in determining whether existing software components are available and suitable to satisfy the functional requirements of a system. Functional requirements can then be established with full knowledge of available reusable components. Management issues regarding reuse strategy, reuse planning, and incentives to promote reuse will be addressed to assist them in incorporating software reuse practices into systems development. Contracting personnel will be given instructions for incorporating software reuse factors into Requests For Proposal (RFP) and Statements Of Work (SOW) and developing proposal instructions, source selection criteria, and award fee plans. Procurement attorneys are provided guidance on how software reuse will affect data rights, copyrights, warranties, license agreements, and patents, so that they can decide the best course of action for the Government. Program control personnel who are responsible for budgeting, costing and pricing will be provided with cost instructions and information to include in RFPs and Contract Data Requirements Lists (CDRL), in evaluating the cost status of efforts, and in determining program-specific costs and benefits, cost estimates, and metrics. Logistics and technical personnel should familiarize themselves with the contents of this handbook to assist in their structuring of specification and work statements.

The concepts addressed assume that the reader has several years of experience in acquisition. Software reuse terminology and concepts are described at a level that is needed to apply reuse during the acquisition life cycle. Guidance in technical areas can be found in the CARDS Engineer's and Component and Tool Developer's Handbooks.

1.1 Software Reuse

From a program management point of view, software reuse can be described based on three functional roles that arise during the reuse acquisition process: supplying, demanding, and distributing components.

Supplying components can refer to developing software components so that they apply to a variety of systems within or between application areas. This development of reusable components can occur within a particular new system development, or by an organization established to support a particular functional area of a using command, as a result of technology transition or commercial software development.

Component demand involves incorporating previously developed software components into a system development rather than redeveloping them. Of course, existing components can be reused as is or modified in some way.

When software reuse is discussed throughout this document, it refers to both developing reusable components (supplying) and reusing existing components (demanding), unless otherwise noted.

A library facility can act as a distribution center to link the two aspects of supplying and using software components. Distributing components pertains to library management activities where software components are acquired, evaluated, tested and sometimes modified. In addition to serving as a facility for the acquisition and distribution of components, libraries usually perform configuration management on the components.

Application areas within which software components are reused are called domains. A domain is a group of related systems that share a set of common capabilities. These domains can be defined as vertical or horizontal domains. A vertical domain is a specific class of system, such as information systems, command and control or weapon systems. A horizontal domain consists of general software functions that are applicable across multiple vertical domains [2] (Vertical and horizontal domains are depicted in Figure 1-1). These can include: user interfaces, common algorithms (data structures, strings, matrices, lists, stacks, queues, trees, and graphs); common mathematical programs (linear systems solutions, integration, differential equations); and software tools or graphics packages.

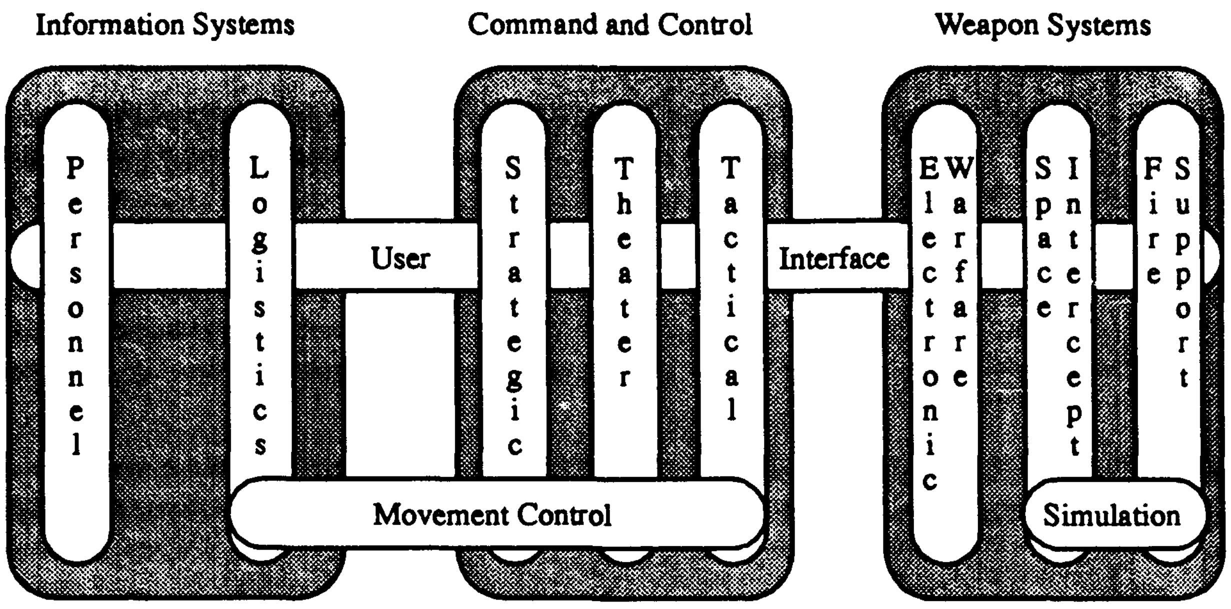


Figure 1-1 Vertical and Horizontal Domains

The first step in determining the reuse strategy and the plans to implement that strategy for a particular system is to analyze the domain. A domain analysis is a process of identifying a domain and collecting, organizing and analyzing data from that domain to determine the common features, whose software functionality is capable of being reused. There are various approaches to domain analysis, but the output is usually a model which represents the application area. This model creates a resource for potentially unlimited future applications [3]. One method is Component Recovery through Re-engineering, which is outlined in Appendix E.

Reuse can be defined, based upon: (1) the granularity of the component, (2) the origin of the component, and (3) the scope or boundaries within which components are reused [4]. Defining software reuse based on these categories can provide a framework for understanding the various ways software components can be reused. Any combination of these categories can become the foundation for developing a reuse strategy for a system.

Component Types

Components for reuse can be categorized into the following: domain model, software architecture, product design and implementation components [1].

A domain model describes the problems within the domain that are addressed by software. The domain model identifies the generic requirements, represents the formal definition of the domain, and provides the general rules and principles for operating within the domain. It indicates the boundaries of the domain, the primary inputs and outputs and the standard vocabulary used [1]. It can be used to communicate desired system features between the user and the developer of a system or group of systems.

A software architecture implements solutions to problems in the domain. It becomes the basis for constructing applications and mapping requirements from the domain model to design components. A generic architecture defines the basic software components, their interfaces, and the means of controlling the execution of the software. It provides a high-level generic design for a family of related applications intended to be reused to meet requirements within the domain. The generic design eliminates the need to develop a high-level design for each application within the domain. As a result, developers use these representations as specifications for reusable components [1].

A product design, which is derived from the specification of the architecture, describes the relationship between the domain model and the work products: it is used to develop reusable components and build systems from such components [1].

Implementation components are at the lowest level and consist of: specifications; detailed designs; code, test related plans, procedures and results: system/software documentation and generated components.

In reusing code, actual code is taken from one application and reused "as is" or modified for use in another application regardless of the system design. It includes both source code (in-house or Government owned) and executable code (Commercial-Off-The-Shelf (COTS) or Government-Off-The-Shelf (GOTS)).

In addition, tools exist that generate actual code based upon user-defined specifications. Some of these tools include capabilities to either prototype the user interface or simulate the system's constraints in addition to generating code.

Component Origin

Components, including entire software packages, can be obtained from the original developer or from reuse libraries. Components are commonly reused by obtaining them from existing Government or commercial libraries. Examples of Government sources include: Asset Source for Software Engineering Technology (ASSET); Defense Software Repository System (DSRS) (formerly Reusable Ada Products for Information Systems Development (RAPID)); Common ATCCS (Army Tactical Command and Control System) Support Software (CASS); Common Ada Missile Packages (CAMP); Central Archive for Reusable Defense Software (CARDS); Reusable Ada Avionics Software Packages (RAASP), and AdaNet. Examples of commercial component packages include: Booch components; Generic Reusable Ada Components for Engineering (GRACE); and the Numerical Algorithms Group Ada Library [5].

Software reuse libraries can be based on vertical domains, horizontal domains, component types, or the software language represented. Libraries can also contain components (e.g., code, specifications, design, documentation...), software tools, or just a listing of component descriptions and sources. A software reuse library based on domain-specific architectures can provide ready access to reusable components by the staff of development and maintenance organizations, and support system composition and rapid prototyping.

Scope or Boundary

The scope or boundary within which reuse is performed can also describe the type and source of components. Reuse can occur: within or across a Program Executive Officer's

(PEO) or Designated Acquisition Commander's (DAC) (or similar counterparts in the logistics communities) spheres-of-responsibility; among services; within systems; across systems within a company; or even across companies.

PEOs especially, and DACs to a lesser degree, are vertically integrated with respect to program responsibility. The current restructuring of services and development organizations resulting from budgetary limitations and the Bottoms Up Review of force structure is creating an environment of change. Some of these changes implement service cooperation and consolidation across systems which fosters reuse. "Lead Programs" within the PEOs/DACs sphere of responsibility could also be established to identify and develop reusable software components. An example is the Joint Advanced Strike Technology (JAST) program, tasked to develop concepts for the next generation of DoD attack aircraft.

However, there is another, higher level of PEO/DAC potential for software reuse, namely across PEOs/DACs rather than within. Identification of a lead PEO/DAC is encouraged within and among the services for subsystems and components which are common among PEO/DAC programs. A current example is avionics for aircraft. The Air Force Advanced Tactical Fighter (ATF) is currently under development. The Navy AX is in the study phase, about to enter Demonstration/Validation. The draft AX Statement of Work encourages contractors to look at the ATF avionics for any commonalities that could provide reuse opportunities. Inherently, software reuse across PEOs may occur here. The "lead PEO/DAC" would be responsible for development and/or identification and subsequent maintenance and improvement of reusable software components in subsystems (such as avionics) which can or do have commonality across DoD service lines. This entire effort is likely to be included in the larger JAST program.

1.2 Reuse in DoD Systems Acquisition

Proper implementation of software reuse can benefit the technical, management, cost, schedule, risk, and maintenance aspects of a system acquisition. These aspects of a system are closely interrelated, so that benefits in one area may be a cause of or result in benefits to another area. Through the use of already proven components, the software is of better quality and more reliable. This in turn accelerates the development and deployment of the system and provide a system that is easier and less costly to maintain during Post-Deployment Software Support (PDSS). In addition, this decreased time will reduces long-term systems acquisition, development, maintenance, and PDSS costs. The technical risks associated with system development can also be identified, managed and reduced at an earlier stage.

Currently, software reuse is being practiced informally, and primarily at the level of code with increasing use at the levels of models, architectures, specifications, and designs, where payoffs are more advantageous. At this informal level, there are no specified reuse methods or processes, and no preliminary analyses are conducted. However, there are some Government and industry initiatives currently focusing on opportunistic reuse, where the methods to identify what types of components may be reused at a given time are integrated into the software development process. The Department of Defense (DoD) is planning to take software reuse one step further to reach systematic reuse, where opportunities are predefined and a process for applying those opportunities is fully specified [1]. To reach this goal and the long-term benefits of shorter schedules, decreased costs and higher quality and reliability, the DoD has developed a Software Reuse Vision and Strategy along with a draft Software Reuse Initiative Program Management Plan and draft Reuse Technology Roadmap. The services have developed Reuse Implementation plans to begin extending the process to the next lower levels. Each of the three services are requiring all subordinate organizations to develop compatible supporting reuse strategies and planning documents.

The DoD Software Reuse Vision and Strategy document focuses on incorporating reuse into all classes of software-intensive systems: information systems, command and control systems, and weapon systems. The plan includes an analysis of the business, technical, and development context in which to implement a software reuse strategy, so that the DoD can move towards a process-driven, domain-specific, reuse-based, technology-supported model for software-intensive development. To support software reuse, the DoD will invest in an infrastructure that centers on advancing technologies that support reuse, incorporating reuse into management and engineering processes, and creating generic sets of components to reuse in new systems and in software maintenance.

1.3 Using This Document

The concepts of this handbook assume that the reader has several years of acquisition experience and is familiar with: the software development life cycle; DoD-STD-2167A - Military Standard, Defense System Software Development; DoD-STD-7935A, DoD Automated Information Systems (AIS) Documentation Standard; DoD Directive 5000.1, Defense Acquisition; DoD Instruction 5000.2, Defense Acquisition Management Policies and Procedures; the Federal Acquisition Regulation (FAR); the Federal Information Resources Management Regulation; and their respective supplements. In addition to these documents, the DoD is changing DoD-STD-2167 to be reissued a MIL STD SDD, which incorporates application of the standard to reusable software acquisition and delivery. A draft of MIL STD-499B, Systems Engineering, also begins incorporating software reuse and "open systems " approaches as "leveraged options" to reduce risk and cost. Focus is on only those topics in systems acquisition that are impacted by software reuse and therefore some aspects of software acquisition may be excluded. The information provided is intended to augment other systems acquisition guidance by providing awareness of software reuse (both developing reusable components and reusing existing components) and its proper application in the acquisition life cycle rather than functioning as a general acquisition guide. The examples and information supplied herein must be appropriately tailored to a program, since software reuse strategies (See Section 2.3, Reuse Considerations for a description of reuse alternatives) and plans for a system depend upon the mission need, goals and objectives of that particular system.

DoD Directive 5000.1, and its companion instruction DoD Instruction 5000.2, provide significant details on the System Acquisition Life Cycle. We do not repeat that information here; rather, our intention in this handbook is to focus on reuse-specific activities. Table 1-1, Reuse Acquisition Process Steps, depicts the process steps and identifies what reuse activities are typically found in the step(s). The parenthetical references identify the specific sections of this handbook which provide guidance for each topic. The items in this table refer to both developing reusable components and reusing existing components.

1.4 Software Reuse Case Study Purpose

Appendix C contains a case study for software reuse. It is provided to illustrate the issues typically addressed in assessing reuse viability and implementing a reuse strategy. The case study shows how the PEO/DAC and Program Manager would become involved, and use the methodologies and templates described in the following sections of this handbook.

Table 1-1 Reuse Acquisition Process Steps (part 1 of 2)

Identify Requirements (User Need)	**Initiate Program Direction and Acquisition Planning**	**Create Solicitation to Reflect Direction and Planning**
• Initiate domain analysis to assess (2.4,2.4.1) Reusabillity Reuse Alternatives • Identify applicable domains (2.4, 2.4.1) Precedented Unprecedented • Initiate cost estimat & schedule activity for reuse (2.4.2, 2.4.3, 7.4) • Initiate cost estimat & schedule activity for reuse (2.4.2, 2.4.3, 7.4) • Identify risks (2.4.4) • Initiate cost estimat & schedule activity for reuse (2.4.2, 2.4.3, 7.4) • Develop/maintain/improve reuse databases Technical Cost (2.4.3) Metrics	• Initiate/continue identification of applicable domains (2.4, 2.4.1) • Initiate/continue domain analysis to asses (2.4, 2.4.1) Reuse viability Reuse alternatives • Identify technical, business, schedule drivers for reuse Use Reuse Consideration table (2.4) Finalize Acquisition Planning (2.4) Identify business issues (2.4, 2.5) Copyright, etc (7.5, 7.9) • Refine cost estimates (2.4.3, 7.4) • Identify risks (2.4.4) • Determine contract type (2.4.5) • Identify reuse cirteria (2.4.6, 6.1) • Define domain architecture approach (2.4.1, 3.2) Government specified or contractor solicited Standards to be used (3.2) • Identify possible component sources (3.2) • Develop/maintain/improve reuse databases Technical Cost (2.4.3) Metrics	• Create reuse requirements documentation (3.1.2) • Finalize evaluation criteria (2.4.6, 7.1) Technical Cost Management Schedule • Finalize proposal instructions (3.2, 6.1) Technical (3.1.2) Management (3.1.1) Cost (3.1.4) Special issues Royalties (3.1.5) Software Rights (3.1.6) • Identify data deliverables (3.1, 3.3, 3.3.2, 3.3.3) • Identify available government information (3.2) Documentation Software components • Develop/maintain/improve reuse databases Technical Cost (2.4.3) Metrics

Table 1-2 Reuse Acquisition Process Steps (part 2of 2)

Evaluation and Selection of Contractors	Program Execution (Design Reviews, Integration & Test and Development Activity)	Operation and Maintenance
• Create and use evaluation standards (4.1, 7.3) • Analyze considerations and checklists for evaluation fo technical and management issues (4.2, 4.3) • Cost evaluation techniques (4.4) • Develop/maintain/improve reuse databases Technical Cost (2.4.3) Metrics	• Use progress metrics in Design Review (5.1, 5.2, 5.3) Technical (5.2) Schedule (5.3) Cost (5.3) Management (5.1) • Assess reuse objectives compliance in test (5.1, 5.2, 5.3) • Provide data for metrics refinement (5.1, 5.2, 5.3) Current program use Future program use • Develop/maintainimprove reuse databases Technical Cost (2.4.3) Metrics	• Assess reuse objectives complianc (5.1, 5.2, 5.3) • Provide data for metrics improvement (5.1, 5.2, 5.3) • Develop/maintain/improve reuse databases Technical Cost (2.4.3) Metrics
Parenthetical references identify sections of the Acquisition Handbook which provide topic guidance.		

2 ACQUISITION/REUSE PLANS

2.1 Reuse Acquisition Planning Structure

The DoD's Software Reuse Vision and Strategy includes requiring business managers to establish plans to manage reuse across their systems [1]. It has been proposed in earlier versions of this handbook, that a DoD organizational structure be established to support the concepts of reuse. This structure includes four functional roles: the Using Organization, those who actually generate the system requirement; the Domain Management Office, which manages reuse across and within systems in a domain and conducts requirements and domain analyses; the Program/ Product Office, which manages the acquisition of the system; and the Software Developer/ Contractor, who designs and develops the system using approved software reuse development standards, guidelines and methods. (Figure 2-1, Reuse Acquisition Planning Structure, illustrates this.) Today, the functions assigned to the domain management office are typically performed at either tne PEO/DAC or program manager level with responsibility determined on a 'case-by-case' basis.

Between the Using Organization and the Domain Management Office, trade-offs (cost, benefit and risk) are made using the domain model output from the domain analysis. Requirements can be initially refined by using high-level components (domain model and software architecture) to prototype the system. These results will determine the requirements of the user and hence the system. Output from this process will be an updated domain model. Hence, between the Using Organization and the Domain Management Office, the *domain model is defined and maintained.*

Subsequently, the Domain Management Office provides the appropriate domain knowledge and requirements to the Program/Product Office responsible for acquisitions. Thus, between the Domain Management and Program Offices, an architecture can be developed based on the domain model.

The Program/Product Office and Software Developer develop the design for a system based on the established user requirements and existing domain knowledge, domain architecture and generic design. These requirements will undergo further refinement through additional prototyping using low-level components (designs and implementation components).

Results of system engineering and development (e.g., designs, specifications, code and documentation) are provided to the Domain Management Office, both incrementally during development and in their entirety upon completion. This assists the Domain Management Office in updating, maintaining and managing the domain model and associated domain knowledge, resulting in availability of domain information for reuse in other systems within and across domains.

To support this organizational structure, top-level planning and policy making guidance should be provided by PEOs/DACs or OSD designated organizations for the following issues: acquisition and contractual, cost considerations, personnel, and technology. This guidance will assist both Domain Management Offices and Program/Product Offices in fulfilling their roles.

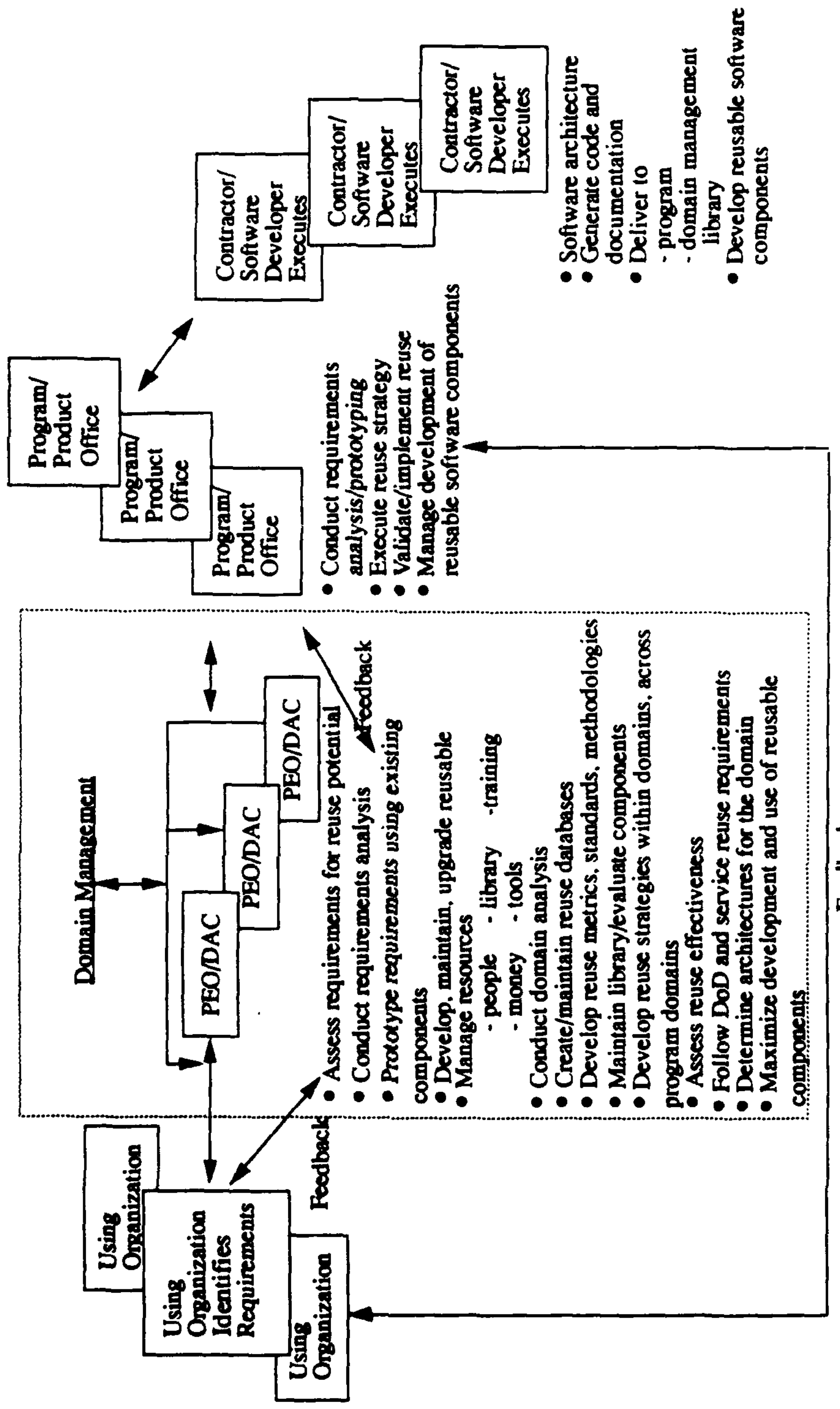

Using Organization
Using Organization Identifies Requirements
Using Organization
Feedback
Domain Management
PEO/DAC
PEO/DAC
PEO/DAC
● Assess requirements for reuse potential
● Conduct requirements analysis
● Prototype requirements using existing components
● Develop, maintain, upgrade reusable components
● Manage resources
- people - library -training
- money - tools
● Conduct domain analysis
● Create/maintain reuse databases
● Develop reuse metrics, standards, methodologies
● Maintain library/evaluate components
● Develop reuse strategies within domains, across program domains
● Assess reuse effectiveness
● Follow DoD and service reuse requirements
● Determine architectures for the domain
● Maximize development and use of reusable components
Feedback
Program/ Product Office
Program/ Product Office
Program/ Product Office
● Conduct requirements analysis/prototyping
● Execute reuse strategy
● Validate/implement reuse
● Manage development of reusable software components
Contractor/ Software Developer Executes
Contractor/ Software Developer Executes
Contractor/ Software Developer Executes
● Software architecture
● Generate code and documentation
● Deliver to
- program
- domain management library
● Develop reusable software components
Feedback

Figure 2-1 Reuse Acquisition Planning Structure

2.2 System Software Development and Domain Analysis

In the context of a typical system software development life cycle, domain analysis is integrated as an essential part of system and software engineering. Figure 2-2, The Classic Software Life Cycle, portrays the software life cycle. Domain analysis would be addressed during systems engineering, and especially during the requirements analysis and design allocation processes to assure software reusability was considered and implemen.ed where practical. Although there are other software development models, such as the spiral process model, the classic software cycle is shown as an example only.

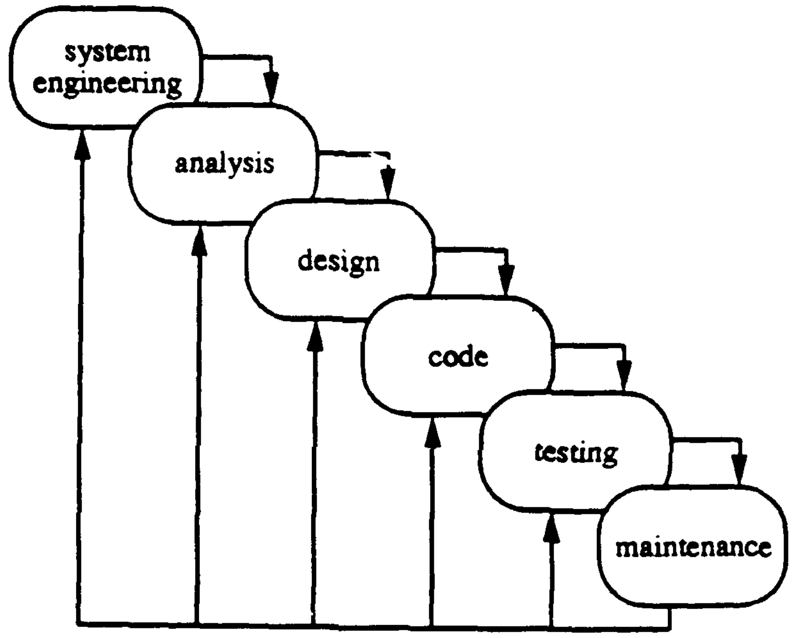


Figure 2-2 The Classic Software Life Cycle (Pressman, 1989)

Figure 2-3, Requirements Specification Development Process, shows how domain analysis supports the system and software engineering process at the program and specification development level. It also suggests that domain analysis must be continually performed at a higher level to support users in analysis of mission needs and operating constraints.

2.3 Reuse Considerations

In analyzing program requirements, there are always technical, management, schedule, risk and budget issues to be addressed and resolved. Table 2-1, Reuse Considerations, is a tool to focus on these five criteria within the context of reuse. These five parameters are integral to the various items within the columns. There is not a one-to-one correlation between each element in the diagram and these parameters, but an example can demonstrate the correlation. "maturity of domains", the first item under the first column "Implementation", is primarily a technical issue. However, there are also cost and schedule implications. How will the analysis of the domains affect the schedule? What are the costs/benefits of performing domain analyses?

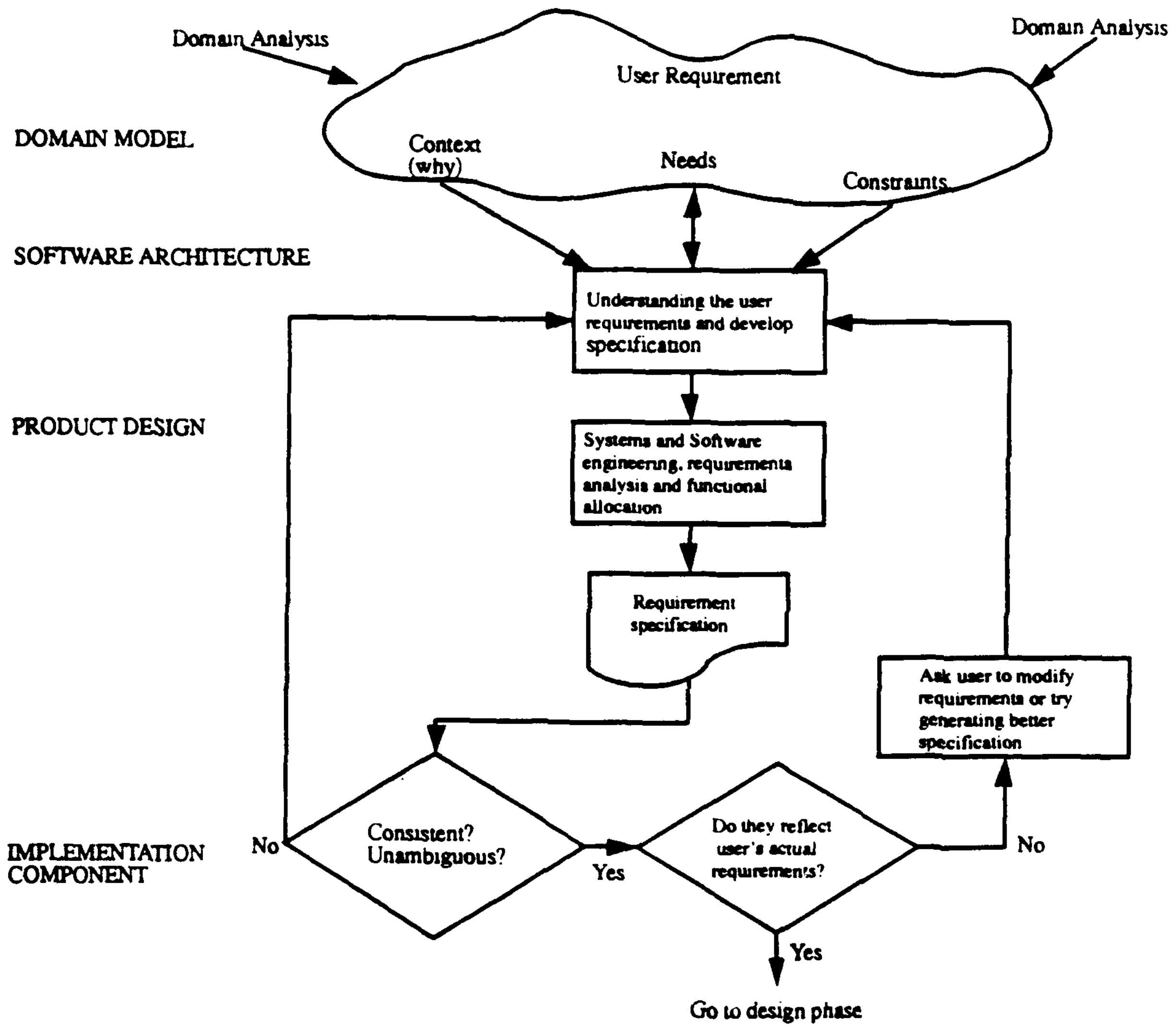

Domain Analysis
Domain Analysis
User Requirement
DOMAIN MODEL
Context (why)
Needs
Constraints
SOFTWARE ARCHITECTURE
Understanding the user requirements and develop specification
PRODUCT DESIGN
Systems and Software engineering, requirements analysis and functional allocation
Requirement specification
Ask user to modify requirements or try generating better specification
IMPLEMENTATION COMPONENT
No
Consistent? Unambiguous?
Yes
Do they reflect user's actual requirements?
No
Yes
Go to design phase

Figure 2-3 Requirement Specification Development Process

The second item within the "Implementation" column, "commitment to resources to investigate reuse" primarily is a budget consideration, but also has technical, management and schedule implications. Needed resources include the correct software tools and techniques, knowledgeable personnel, and appropriate levels of funding.

Table 2-1, Reuse Considerations, provides information to assist in:

- Assessing the viability of reuse in an acquisition. Again, determining reuse viability includes both deciding which components will be developed to be reusable and which will be reused (with or without modification) from existing components.
- Determining the reuse strategies and approaches which make the most sense
- Assessing whether sufficient resources exist to pursue reuse (these resources include: technical support, trained personnel, financial commitment, administrative support, regulations, policies, and management commitment).
- Determining the best way to implement the strategies, given the availability or subsequent acquisition of resources.

Table 2-1 should be utilized in light of the specific reuse alternatives identified below:

- New (Unique) - Unique Development (no reuse)
    - A software component is developed for unique purpose and will not be used again
    - No reuse of the component can be identified at this time
- New (Reuse) - Develop to be Reusable
    - A reusable software component is developed to satisfy a particular need, but can be reused for other than its initial purpose and intent or it is specifically developed as a reusable component.
- Reuse (Modified) - Modify Existing Components
    - Existing software components are modified for reuse (includes upgrade of supporting documentation)
    - These components include:

      COTS/GOTS: adding device drivers or wrappers

      Public domain software

      In-house software
- Reuse (As Is) - Reuse Existing Components As Is
    - Existing software components are reused as is

- Existing (non-commercial) reusable software components, such as GOTS and/or contractor non-development software (i.e., it exists but has not been commercialized or otherwise used) is reused "as is". No modification or significant upgrade of supporting documentation is required.

- COTS (e.g., Lotus 1-2-3, DeLorme Mapping System) is purchased and used as is, as part of a system.

- Any combination of the above.

Table 2-1 Reuse Considerations (part 1 of 2)

Implementation Plan	Risk	Management Support	Regulatory/ Managerial Considerations	Documentation
• Maturity of domains Precendented vs. unprecedented • Commitment of resources to invetsicate reuse Quality and experience of personnel Time • Existing technology base • Existing resource base • Mandate domain analysis • Availability of metrics • Communications betwee user and PEO/DAC on domain model • Communications between PEO/ DAC and Program Manager on architecture	• Technical Lack of libraries & tools Ability to define reuse objectives Maturity of reuse system engineering process Inadequate documentation • Schedule People Models • Cost Budget availability Cost model availability • Metrics availability • Lack of understanding of reuse requirements • Knowledge and experience level of personnel Commitment & advocacy	• Funding support • Implementation authority • Availability of metrics • Communication between user and PEO/ DAC on domain model • Communication between PEO/ DAC and Program Manager on architecture • Provision of adequate training and education • Changes in methodologies • Changes in Acquisition Policy	• Rights Ownership levels Copyright • Recoupment • Clear policies on reuse goals • Adaptability of policies & regulations to support reuse objectives • Availability of metrics to assess reuse goal progress • Communication betwee user and PEO/DAC on domain model • Communication between PEO/ DAC and Program manager on architecture	• Reuse strategy plans • Metrics reporting • Reuse development plan • Support for changes in methodologies & documentation • Test & refine feedback • Proposal data requirements evaluation • Adequacy of software/system documentation

Table 2-2 Reuse Considerations (part 2 of 2)

Strategies	Initial Investment	Cost	Incentives	Schedule	Liability
• One time vs. multiple reuse • Need for Government maintenance capability • Level of rights ownership required • Commercial product potential • Industrial base maintenance/ expansion • If Government ownership, who maintains	• Technology support for reuse initiatives • tools • methodologies • databases • Domain analysis • Domain mgr structure • Libraries • Training & education costs • Technology transfer • Library population	• Develop reusable components • Use existing reusable components "As is" or modified • Cost of incentives, royalties, license fees, award fees • Library operation & maintenance	• Government Cost & schedule potentialsavings Lower risk career motivation • Contractor Financial return on investment Commercial product competitive advantage	• Impact on program schedule • Library search • Developing for reuse • Documentation & test for reusable components • Parallel development	• Need for warranty coverage • Performance • Quality • Third party responsibility

Reuse Considerations are numerous across a wide range of policy areas.

The reuse alternatives are be assessed in the context of these considerations. Taking the "Strategies" column and its first item, "one time versus multiple reuse", as an example, the process follows:

<u>Consideration:</u>	<u>Strategies</u>
Item:	One time versus multiple reuse
Applicable to program:	Yes, applicable. Multiple reuse applies.
Reuse Alternatives	
New (Unique):	A unique development is not applicable, since reuse is viable (validated by domain analysis).
New (Reuse):	Viable approach. Offers opportunity to exploit reuse functionality in current and future programs in similar domains

Reuse (Modify):	May be effective if domain analysis and library research validates existing, available GOTS or other existing components. Probability of use "as is" for this program is low
Reuse (As Is):	COTS - Insufficient knowledge at this time

This process would continue for each consideration and the items contained in the considerations columns The questions to be asked against each item will vary from program to program as the concept of reuse and objectives for it are defined. However, the process supports a decision on the viability of each reuse alternative and also identifies the constraints and parameters involved in adopting reuse in a particular program.

Throughout this handbook, additional matrices, tables, and decision models are provided for use in formulating questions and assessing these reuse considerations.

2.4 Preliminary Planning

To effectively incorporate software reuse into the acquisition planning cycle, reuse must be considered at the beginning of the process. The software reuse strategy should be developed concurrently with an acquisition strategy. The software reuse strategy is a product of the domain analysis. When software reuse, as a process, is better established, the reuse strategy will be in place prior to the systems acquisition actions. A formal Reuse Strategy document for a system can be a separate document from the Acquisition Plan, but it is recommended that the two are contained in one document and address all of the issues identified in this section. Present DoD and service policy require a Software Reuse Implementation Plan at the PEO level, and the policy is being extended in some organization responsibilities. Though the documents are more general than a reuse strategy for a domain or specific system, they do implement the process to establish formal planning for reuse to the lowest level of acquisition in the DoD.

Since the Acquisition Plan implements strategy formulated and agreed to by the Acquisition Strategy Panel, it is imperative to include experts in both domain engineering and the application domain area(s) on the panel. Also, when the program office is established, personnel must be educated and trained on the business implications of reuse and the technical methods available. The proposed Reuse Strategy section of the Acquisition Plan becomes the top-level planning document for reuse within a system acquisition. It ensures the effective integration of the various reuse methods and techniques with acquisition life cycle events, documents, and activities. In developing the Acquisition Plan and Reuse Strategy for a system, the DoD Software Reuse Vision and Strategy and the appropriate service-related reuse goals, policies, and plans must be followed.

The feasibility of incorporating software reuse into a system acquisition is assessed by evaluating and analyzing the possible impacts on the system's goals and objectives. Software reuse may have a tremendous impact, either positive or negative, on the strategic, technical and life cycle concerns of the program. The decision to implement reuse depends not only on these effects, but also on resource availability. These resources include: knowledgeable program management personnel, time/schedule, cost/funding, business practices, the technical capabilities of both Government and industry, and existence and access to the domain knowledge and components.

This section addresses the process to follow and the issues to consider in determining a software reuse strategy for incorporation into the acquisition strategy. It assists in determining whether reuse can be accomplished, the impacts on a system, and if and how to implement reuse. First, the guidelines for developing a program's technical, schedule and cost baselines are addressed. Then, based on these baselines, considerations for estimating costs and determining the budget are discussed. Finally, the appropriate contract type can be determined and the evaluation criteria developed.

Appendix B contains a checklist for contract considerations for reuse, addressing technical, cost, rights, data and other topics impacted when structuring reuse strategy.

2.4.1 Program Technical Baseline

The Defense Information Systems Agency (DISA) has identified and described the domains and sub-domains of software systems in the DoD. Each domain is characterized in terms of common functions shared by the different systems. The results of this study become a valuable resource for PEOs to incorporate software reuse into systems acquisitions.

PEOs and DACs are encouraged to establish plans to manage reuse across systems within their spheres-of-responsibility. Within these spheres-of-responsibility, domain analyses can be conducted and a domain-specific library established and maintained. Whether a domain analysis must be performed or the results of a domain analysis evaluated, depends upon domain maturity. The following domain analysis discussion refers to both conducting a domain analysis and analyzing the results.

Determining the technical baseline for a system assists in developing a software reuse strategy. The reuse strategy depends upon the mission and objectives of the system, advancement of the technology within the domain and expertise in the field. The domain is analyzed in light of program objectives and requirements. Table 2-2, Reuse Technical Baseline, shows some issues to consider before determining the reuse strategy.

Table 2-3 Reuse Technical Baseline

ANALYZE DOMAIN					
Domain Boundary	**Domain Status**	**Component Requirements**	**Component Analysis**	**Libraries**	**Technical Resources**
Does system belong in an existing domain? Does that domain boundary encompass all system requirements?	Mature Understood Stable Predictable technology Demand for components Supply of components (existence and quality) (Note: components refer to domain models, generic architectures, product designs, implementation components.)	What component types will be needed: domain model, software architecture, product design, implementation components? What descriptors are needed: author, description, domain, cost, legal, security restrictions, source language, size? Additional component data required? • test tools and results • design trade-off analysis results •, quality tools and analysis	What is demand for components in domain? • What other systems will need these components? • What are their requirements? What are implications of depositing into library? (additional testing, impacts on rights?) Who has owner ship? • Government, commercial What are costs to: • obtain? • modify? • test? • assess certification level?	Are existing libraries applicable to domain(s)? Are the libraries accessible? hardware/ software compatible, interoperable What procedures and processes does library require to access or to provide data components? How will procedures affect the program technical baseline? What are the evaluation criteria for the library to accept components? (quality, testing, documentation, support)	Tools, methodologies,processes Domain Engineering experts Metrics Domain experts

ANALYZE DOMAIN					
Domain Boundary	Domain Status	Component Requirements	Component Analysis	Libraries	Technical Resources
			Who holds data rights; what are implications? Components easily accessible: interoperabe? Components relevant to application? Components applicable to specifications? Must components be modified? • to what extent? • is it cost effective?		

Those involved in the acquisition and reuse planning of a system must have a clear understanding of and familiarity with the objectives of the program, the technical requirements of the system, and the domain considerations. These objectives will assist in establishing the bounds of the technical baseline and become the foundation for determining the reuse strategy.

Some examples of program objectives include: no new development, minimal development, low-risk system, particular date specified for the Initial Operating Capability (IOC), or the system must contain state-of-the art technology. For example, reusing existing components should be investigated when a program's objective is one of the following: "no new development", "low-risk system" or "a near-term IOC date is specified". In these cases, developing reusable components may not be a choice. If a program objective is "contain state-of-the art technology", then the development may be a unique one and no reuse applies. On the other hand, the objective could be "contain state-of-the art technology", but it may not be unique. Thus, the application would apply to other systems and aspects of the system could be developed to be reusable.

A domain should be analyzed for its effect upon anticipated systems development to determine when existing components will be reused during a development, components will be developed to be reused, or a reuse library will be developed. There are six major areas of the domain that should be investigated: the domain boundary, domain status, component requirements, component analysis, libraries, and the availability of technical resources. Table 2-2, Reuse Technical Baseline, gives a checklist for examining these issues.

The system being planned may fit into more than one domain; thus, the boundary of the new system's domain must be carefully defined to encompass all applicable domains. The goal is to work within a domain which is well understood, mature and stable, and utilizes predictable technology. Other aspects that can be examined to determine the maturity and stability of domains are: supply for components, demand of components, and the existence and quality of domain models and generic architectures.

Components can be examined from two points of view, examining existing components and the system's requirements. The analysis/research can bring to light whether or not there are components in existence that can be used and the technical implications for using them. For example, is there a future need for the types of components you will develop? The present and future demand for components in the application domain(s) must be estimated. Do existing components fit into the program/system specifications? If not, the extent of modification must be estimated. Examining the system's requirements reveals the type of components needed, as well as additional data explaining the components and levels of acceptance criteria needed before any component is considered as a solution to systems requirements.

Are there libraries in existence that cater to the particular domain of interest? If so, are these libraries accessible, in terms of both software and hardware? Are the acceptance criteria for the library more or less strict than those for the system? It must be determined whether components need to go through additional testing if deposited into libraries. If so, these additional tests must be incorporated into the development cycle.

Finally, an important aspect is the availability of technical resources. What processes, tools and methods are needed to incorporate software reuse into the life cycle and are there any in existence that satisfy these needs? Are there experts available who are familiar with the particular domain area and may be used to facilitate domain analysis and domain engineering?

Once each of these aspects is compared with the requirements, they can be matched to the particular reuse alternative discussed in Section 2.3, Reuse Considerations, to assist in determining the overall reuse strategy.

2.4.2 Program Schedule Baseline

The overall program schedule is a natural outgrowth of the technical baseline. The program objectives discussed in Section 2.4.1 drive the length of the program schedule, and the reuse strategy adopted has a direct affect on the time needed to design, develop, test, produce, deploy and support the system. Whether software is developed to be reusable or existing components are reused, there will be positive impacts on the schedule if these components are high-level components (domain model, software architecture).

Developing software specifically for reuse may involve additional time over the normal development schedule for software that is to be used one time only. Software developed for reuse needs to be generalized, handle a wider range of input parameters, offer acceptable performance in most cases, and handle unexpected and erroneous inputs and conditions better than one-time software. Designing and building software to satisfy these criteria may involve extra development time, as well as an increase in testing and documentation of the components to support future reuse. For software components to be truly reusable, they must be subjected to rigorous testing to ensure quality (and reduce risk to the subsequent user), and must be well documented to facilitate ease of integration by the subsequent user. These activities must be done when the components are first developed (subsequently, the domain management library should control them), and will add to the overall program schedule. Program managers must be supported by PEOs/DACs in accepting any schedule (and cost) impact associated with developing software for reuse. If the program objectives cannot accommodate the additional time required, then developing reusable components may not be a feasible option. We should note there is no evidence of any significant cost or schedule impact associated with developing components to be reusable. There is limited data available today but most would agree that "significant impact" is not an accurate assessment.

In these cases where the technical baseline for the program may specify "no new development", "low risk system" or "a near-term IOC date specified", and reusing existing components is judged to be a worthwhile objective, the impacts on schedule again have to be examined. First, there is some up-front time devoted to investigating sources of reusable components. This is time spent over and above the normal development schedule to search out libraries within the domain, analyze the accessibility of the libraries, and validate the relevance of the components for use in this particular program. This search time is a function of the number and quality of sources of reusable software for the application at hand, as well as the capability level of the personnel doing the search and validation. However, once the reusable components have been identified, validated, and accessed, there is a positive effect on program schedule, in that the reused components do not have to be redesigned and recoded (they, however, still must be integrated and tested with the rest of the software). The amount of time saved is a function of how many components are reused and how complex those components would have been to develop from scratch. If the time spent in finding and validating reusable components exceeds the time saved by not having to redesign and recode the reused components, then the decision to reuse existing components may not be a good one. If, however, there turns out to be a net savings in development time due to component reuse, then program objectives may be satisfied and the reuse decision may be a good one.

The wider question of where the particular components fit in the domain software architecture is another issue. As domains become better organized, specific functions can be assigned to individual programs to prepare a reusable component for their system with the further requirement to be reusable across the domain. Consideration of cost and schedule becomes more complex due to the impact on the entire domain program.

The need to investigate the impacts of software reuse on schedule is not isolated to the design, development, and test phases. There are also effects on post-deployment schedule that must be considered. For instance, time allotted for training could be reduced due to similarity of components in different systems.

2.4.3 Program Cost Baseline

<u>Estimates</u>

Software development cost estimation techniques generally fall into two main categories: 1) parametric models which use lines of code or function points and other parameters/factors to predict development cost, and 2) productivity metrics which predict development cost based on the number of lines of code developed per day or the cost per line of code. Whatever cost estimation technique one uses, it must be remembered that the databases supporting these techniques generally do not include software reuse as a normal occurrence. In other words, these techniques were developed with the view that all software is newly developed and software reuse, whether reusing existing components or developing software for reuse, is a condition that requires the technique to be adjusted.

An experienced user of a software cost model can simulate the presence of software reuse by adjusting certain input parameters. If software components are being developed for reuse, the analyst can upwardly adjust the relevant parameters that drive up the testing and documentation costs of components being built for reuse, but the process is, by nature, imprecise. The accuracy of the estimate is a function of the experience of the analyst and his/her ability to adequately portray the expected increase in the adjusted parameters. Without significant data from reuse programs to "feed" this process, the results can be suspect. Therefore, it becomes imperative that PEOs/Domain Managers establish databases within their domains to capture relevant cost, schedule and technical data on historical reuse programs to counter the current estimating deficiencies.

Similarly, for the case in which existing components are reused, the analyst can reflect the expected decrease in designing and coding costs by downwardly adjusting the parameters which directly affect the costs for those activities. Again, this is not a scientific process backed up by hard data, but rather an instinctive process that can result in a highly inaccurate estimate of program costs. Clearly, builders of parametric software cost models must be challenged to collect the relevant data and include software reuse considerations (both development of software for reuse and reuse of existing software components) into their models.

Cost estimation methods are also required for higher levels of components and software-related activities (i.e., not just code, but also domain analyses, domain modeling, architecture development, and other higher-order efforts which must be performed before estimates of reuse costs can be attempted). Activities like domain analysis and architecture development have not been cost modeled in the past, since they are more appropriately handled by traditional, bottom-up accumulation of labor and material costs associated with the effort. This does not mean, however, that reuse models cannot be built which incorporate costing for both software components and higher-order activities.

In the same context, the currently-collected software productivity metrics focus only on the one-time development of software and do not specifically look at reuse. Most software development organizations maintain metrics which show that they can develop some number of lines of code per analyst per day. What if some or all of the software in the next program is being developed for reuse? Can the metrics be easily adjusted to reflect the expected lower productivity (due to more extensive design, additional testing and documentation) when developing for reuse? For the organizations which maintain metrics that define the cost per line of code, can the metrics be adjusted to reflect the expected higher cost per line when developing for reuse? The answer to both questions is probably "no". The problem with productivity metrics also exists when looking at the case of reusing existing components. The metrics cannot be easily adjusted to account for the higher productivity or lower cost per line of code when reused components are a part of the program. Once again, the collectors of software productivity data must be made aware of the shortcomings of current metrics to adequately estimate the costs of programs which contain reuse.

A good source of available metrics is the Software Engineering Institute's (SEI) technical reports on measurement [6], [7], [8], [9], [10]. Today metrics specifically designed for reuse are not available. Rather, existing metrics must be fine-tuned to account for any actual reuse experience and/or projected performance.

The same problems which exist with current software development cost estimating techniques are also present when looking at software maintenance costs. The maintenance models and metrics were developed without any consideration for software reuse issues and, consequently, must be adjusted to account for reuse. In short, current techniques for estimating software development and maintenance costs do not accurately portray the costs associated with developing for reuse or reusing existing components. New techniques which have been developed specifically for reuse are needed, in particular to guard against overly-optimistic estimates of reuse savings which can result from inadequate software engineering analysis. For now, the Program Manager should require fine-tuning of existing models to recognize anticipated maintenance experiences derived from reusable software (e.g., savings in training and maintenance documentation and actual training and maintenance required.)

Programs which contain reuse may entail other cost estimating considerations not normally included in one-time development programs. If existing components are to be reused, there may be some form of incentive paid to the current developer for searching out, validating, and incorporating reusable components into the overall architecture. Also, the original developer(s) of the components contributed to a reuse library might receive some financial reward (e.g., license fee) each time those components are reused. These are "over and above" estimating considerations peculiar to reuse that cannot be overlooked. While these concepts are potentially controversial, since many disagree that developers must be incentivized, they nonetheless represent alternatives which have cost impact if utilized. SAF/AQ Policy Memorandum 93M-007, Software Reuse Incentive Policy, supports active exploration of incentives to implement software reuse policies in contracts.

Section 7.4 contains a simple cost/benefit model framework that will assist in making reuse decisions. An Example of how this framework might be used is provided below.

Establish the cost of the software development strategy, with and without reuse, to assess whether reuse makes economic sense.

If:

Scn_1 =	Software development costs of new software not developed for reuse (entire system)

Scn_2 =	Software development costs of new software not developed for reuse (portion of entire system)
Scr =	Software development costs for reusable software
Scm =	Software development costs for software modified to be reusable
Scs_1 =	Software used "as is" (NDI) for new development without planned reuse
Scs_2 =	Software used "as is" (NDI) for reuse approach
Sco_1 =	Operation and maintenance costs for software not employing reuse
Sco_2 =	Operation and maintenance costs for software employing reuse

Then, software reuse becomes economical for an individual program when:

Cost of software reuse development and maintenance < cost of a unique (no software reuse) development and maintenance

$$Scn_2 + Scr + Scm + Scs_2 + Sco_2 < Scn_1 + Sco_1 + Scs_1$$

Software reuse is economical in assessing multiple programs when the costs to develop reusable software are offset by costs avoided in those other programs.

If the costs of new software component development versus reuse are approximately equal, we can reasonably anticipate that life cycle cost considerations/savings favor reuse in terms of reductions in training, increases in reliability, reduced costs of future modifications and similar factors. If reuse costs are greater than new development cost, then savings in these other life cycle factors must more than offset the increased development costs.

The inputs to the cost model vary with chosen strategies or available alternatives. For instance, one conclusion of a domain analysis may be that there is/are no software component(s) available for modification for reuse. In that situation Scm = 0. Similar variations could occur with technical and/or schedule decisions regarding the levels of reuse (e.g., architecture versus code).

This model is very top-level and is only intended to provide insight to the issues that must be considered when assessing reuse economic viability and its associated costs. The more detailed cost models which estimate new software development costs (e.g., COCOMO and SEET) and models becoming available to estimate reuse costs (e.g., Kang/Levy (SEI) and Gaffney/Durek (SPC)) must be employed to develop the detailed costs which support this top-level model.

Budgeting

Assuming that one has made a reasonable estimate of the likely costs for a program which incorporates a specific degree of reuse, the next step is to establish the program's budget. The budgeting process determines how much and what type of dollars, by fiscal year, are needed to support the program. Overlaying the cost estimates onto the program schedule identifies when the funds are needed. Service regulations govern what type of dollars (e.g., Research and Development (R&D), Procurement, Operations and Support) can pay for which activities. Even with the regulations, however, there are issues for a program which contains reuse that must be clarified. For example, it is clear that, when developing software for reuse, the additional costs for added design, testing and documentation should be included in the R&D budget. Similarly, if a development program is using existing components, and the developer is being incentivized to search out and use existing components, those incentives are properly charged to R&D. However, if a development program is reusing existing software components, and there is a license fee to be paid to the original developer(s) of the component(s), should those fees be paid for with R&D funds, or with procurement or some other type of funds? This and other budgeting questions, unique to the reuse case, must be addressed. Typically, one expects Government obligations for payment to be made out of program funds for the program phase when the obligation is actually incurred. For instance, a royalty is paid when you use the component, not when it is developed. The use of any and all of these funding categories could be interpreted as appropriate, under varying conditions. R&D funds should be used to develop reusable components or to develop systems which include reusable component acquisition. R&D or procurement funds should be used to actually obtain the reusable components. Operations and Support (O&S) funds should be used to maintain the systems, including any reengineering and/or software component upgrades into reusable components or architectures. See Table 2-3, Reuse Funding Alternatives.

Table 2-4 Reuse Funding Alternatives

	REUSE ALTERNATIVE			
TYPE OF FUNDS	Develop reusable software component	Acquire reusable component	Modify compo- nents to be reusable	Maintain reuse components
R&D	X	X	X	
Procurement		X	X	
O&S		X	X	X

2.4.4 Risk Assessment and Mitigation

Software reuse can impact systems acquisition since it is a new technological process, and has yet to be implemented DoD wide. Thus, when assessing risk and developing the risk management program, the technical, schedule, cost and management factors that are affected by reuse must be examined. As in any other software acquisition, the risk status must be continually monitored and evaluated to determine actions for mitigation. However, in software reuse development, risk evaluation and monitoring must be more intense and occur more often.

The primary factors that may impact the technical aspects of a program relate to requirements, software development, documentation development, and software performance. Since software reuse is a new process, software designers and developers may not have a complete understanding of the reuse requirements. There are software development risk implications for developing reusable components, reusing components as is, and modifying existing components. Systems engineers tend to have a one-system view as opposed to a more global view. Since reuse is so new, there may not be appropriate domain libraries available for both depositing and acquiring components. Also, there may be a risk of software reuse support tools not being compatible with a particular software engineering environment. If the domain is not scoped and analyzed properly, components may become obsolete. Documentation may not be developed to apply to the generic components and thus may not be reusable along with the component. Existing components may not meet the performance criteria of the new system under development. But even if they do meet the criteria before integration, subsequent integration may impact other aspects of the system and the system as a whole may not meet the specified performance criteria after integration. In addition, systems within a particular domain often have different performance requirements on critical areas. These divergent performance requirements may be domain functions that do not have commonality across systems, and thus, these components may not be able to be reused.

Designing generic components and acquiring existing components may impact both costs and the program schedule. Cost and schedule risks relate not only to the technical issues, but also to management issues. Management plans for reuse must be sufficiently flexible to allow incorporation of new technologies and methods. There also may tend to be a lack of communication, and cultural issues might arise that the managers must properly address to ensure success of reuse initiatives.

The technical, cost/schedule, and management risks for software reuse and associated mitigations are listed in Tables 2-4 through 2-9 [11]. These should be considered for all phases of the software life cycle: development, maintenance, modification, and support.

Table 2-5 Technical Risks and Mitigations: Requirements

RISK	MITIGATION
Lack of understanding of technical and reuse require ments	Conduct domain analysis at PEO/DAC level for all potential systems to be developed Use domain experts during requirements specification phase Base Requirements on the domain model and software architecture Conduct an initial assessment of all reusable assets and identify any reuse implications (including emerging COTS products) Match required and available resources Model and prototype system and user requirements to identify reusable components Use prototyping to validate/refine reuse approaches Assess the impact of any proposed changes (systems and software) on reuse goals/strategies Carefully document reuse requirements in system spec ification documents Analyze reuse requirements trade-offs Analyze and assess new technologies, techniques, and methods to support reuse
Lack of knowledge about domain analysis concepts	Provide training, education, and management support
Lack of domain models	Conduct domain analysis at PEO/DAC level for domain and identify potential systems to be developed
Inadequate representation mechanisms for requirements specifications and domain model descriptions	Carefully document reuse requirements in systems specifications

Table 2-6 Technical Risks and Mitigations: Software Development

RISK	MITIGATION
Lack of reuse components and/or libraries	Perform analysis for library definition and creation (per PEO/DAC) Propose standards to promote reusable components. Set up & maintain library (need to acquire and disseminate components)
Lack of appropriate support tools	Research current reuse support tools Develop support tools Acquire appropriate support tools
Components will become obsolete	Design for open architectur Plan for PrePlanned Product Improvements Include technology assessment during domain analysis
Inadequate systems engineering view on reusability: parochial versus global perspective	Investigate and exchange trade studies and prototyping to promote reusable designs Establish design criteria and standards to be used in design reviews and trade studies Perform independent validation with organization to ensure a global reuse perspective Devise specific designs and tests to validate reuse func tionality Prototype reuse functionality in other than baseline sys tem (Preliminary simulation, laboratory facilities) Conduct a limited demonstration/ validation of reus ability of that functionality Provide education and training Provide support from domain analyst

Table 2-7 Technical Risks and Mitigations: Documentation

RISK	MITIGATION
Documentation not developed with reuse in mind (generic)	Develop clear, concise documentation Standards and mandate their use Provide education and training

Table 2-8 Technical Risks and Mitigations: Performance

RISK	MITIGATION
Improper integration of reused software components	Conduct: Analysis and trade-offs Performance modeling & benchmarking Regression testing Simulation Use domain models and generic architecture Proper choice of reusable components
The software components will not meet performance criteria	Fine-tune components to improve performance Proper choice of reusable components Incorporate accurate reuse software benchmarks into performance model (from previous use) Prototype Explore scalable designs Measure individual performance of reuse components (NDI), newly developed components)

Table 2-9 Cost/Schedule Risks and Mitigations

RISK	MITIGATION
Increased costs and lenthened schedule due to designing, coding and documenting generic components and/or reusing existing components.	Properly train staff Hire experienced staff (reuse and domain experts) Validate cost models designed specifically for reuse Validate cost model inputs

Table 2-10 Management Risks and Mitigations

RISK	MITIGATION
Inadequate management support	Provide proper reuse training Allow/enhance communication in both directions Establish reuse objectives as part of the performance reviews for team leaders Commit to incorporating reuse (provide incentives to personnel) Select reuse champion to advocate reuse
Inadequate reuse planning for adaptability	Ensure coordination between program personnel within an application domain Use domain models and software architectures Conduct preliminary reuse planning Incorporate reuse and risk management strategies Track configuration management (reuse) Identify reuse policy and goals Acqure tools and support environments (include in planning & budgeting)
Poor communication/cultural differences	Plan/conduct technical exchanges and management/ staff meetings (document meetings, follow-up) *Provide technical support by reuse specialists during the implementation of the first use of reuse technolog* Provide education and training in reuse technology Identify nature of cultural issues (with assistance of objective third party) Assess and implement measures required to effect desired changes in attitudes and behavior Minimize resistance to change by implementing reuse in progressive steps (not at a rapid rate)

2.4.5 Contract Types

Selection of contract type is a function of risk. While other factors or strategies may influence selection, risk must be the predominant factor. Risk to a contractor (and similarly to the Government) involves: (1) Stability of the requirements as detailed in the specification; (2) extent of the program's technical, schedule, and cost risk; and (3) levels of anticipated or required Government involvement. In general, cost reimbursement type contracts are recommended as program risk increases and fixed price type contracts are recommended as program risk decreases (See Figure 2-4).

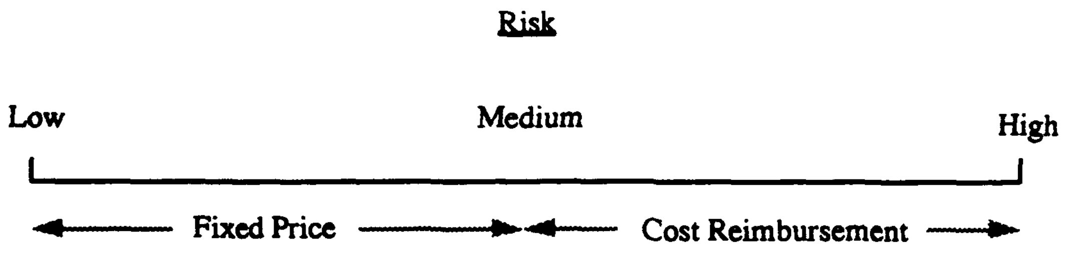


Figure 2-4 Program Risk and Contract Type

Similarly, as the degree of Government involvement moves from low to high, the same pattern for selection of contract type occurs (See Figure 2-5).

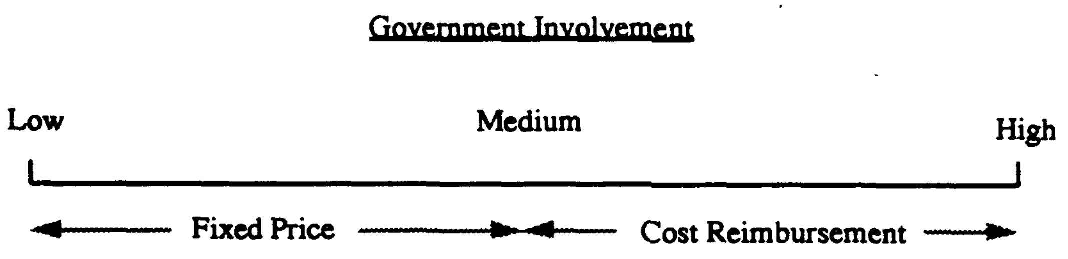


Figure 2-5 Government Involvement and Contract Type

The reasons for these phenomena are simple. As program risk (uncertainty) increases, the contractor cannot reasonably be expected to be able to estimate contract costs with any high degree of certainty. Similarly, as Government involvement increases, the contractor has less ability to manage and control its work and costs. Cost reimbursement then becomes more appropriate. As risk and Government involvement decrease, we can move towards fixed price contracts.

Section 2.4.4 discussed the impact of software reuse on program risk. We would expect to move towards cost reimbursement contracts when developing software to be reusable, and fixed price contracts as we reuse large amounts of software components and are primarily involved in integration and test (see Figure 2-6). However, reuse may be only one component of overall program risk, and has to be considered in context with all other risks. We cannot as easily anticipate the "Government involvement" impact associated with reuse. This "Government involvement" is typically associated with uncertainty in requirements and architecture for reusable components potentially requiring more intensive Government participation in requirements analysis and architecture decisions. While program risk might be low (inclusive of reuse), the Government may choose to have significant visibility (i.e., level of interaction) into the reuse effort. This is legitimate, but it impacts the contractor - the more significant the involvement, the more we move away from fixed price towards cost reimbursement contracting.

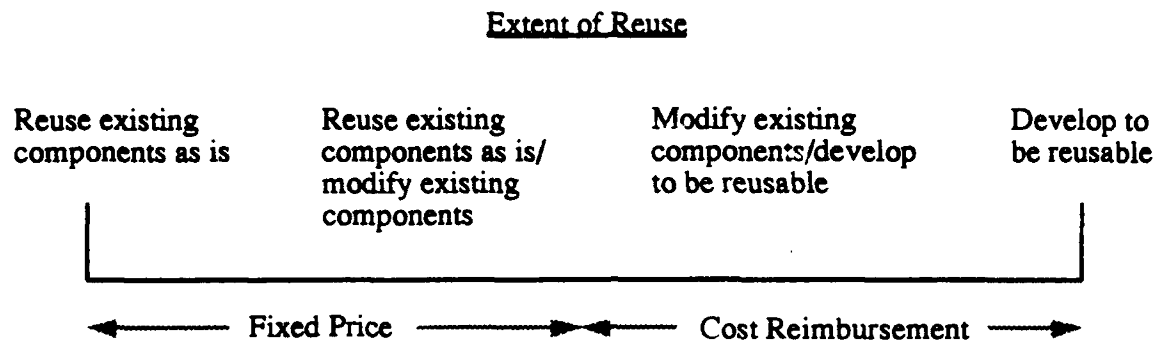

Extent of Reuse
Reuse existing components as is
Reuse existing components as is/ modify existing components
Modify existing components/develop to be reusable
Develop to be reusable
Fixed Price
Cost Reimbursement

Figure 2-6 Extent of Reuse and Contract Type

Of course, variations on these relationships and strategies occur. Fixed price incentive contracts with higher ceilings (perhaps greater than 120%) and more shallow share ratios (80/20; 90/10: Government/contractor) can provide more cost protection to the contractor under either increased reuse risk and/or Government involvement. Alternatively, contract types could be mixed, with cost reimbursement or some fixed price variation (as just described) used for higher risk/involvement reuse tasks, and fixed price type used for the balance of tasks if they are of lower risk/involvement.

Selection of contract type for software reuse must also consider shifts in responsibility. The more responsibility the contractor is expected to assume to lower his risk, the more we tend towards fixed price contracts. High contractor responsibility and high Government involvement (or high reuse risk) are incompatible. One cannot ask a contractor to assume all reuse task responsibility under a fixed price environment and still expect to have significant degrees of Government involvement; the concepts eventually become mutually exclusive.

Remember, selecting a good contract type will enhance program reuse objectives; a poor selection will not solve program problems and most certainly will contribute to, if not cause them.

Incentives are extensively discussed in section 2.5.6. We would note here that if incentives are to be used for reuse (award fee, cost incentives, etc.), they must be balanced with other contract incentives to assure each incentive provides continuous, positive motivation. A classic example of two incentives used is where an incentive (positive/negative) exists to control costs and another to motivate faster deliveries. If the delivery incentive provides greater rewards than the negative cost incentive can inflict, the contractor will typically incur the cost penalty to make earlier deliveries. The Government would then incur added costs and still be obligated to pay the delivery incentive. Similarly, if developing reusable components were critical, an award fee might be used to motivate development that results in high quality and functionality of components. If the reusable component development were performed under an incentive type cost reimbursement contract (CPIF), one would ensure that the cost incentive parameters (Government/contractor share of overrun; fee structures) were not so skewed that the contractor focused on management of costs to the detriment of reusable product development. Again, balance is necessary - perhaps, in this example, a simple cost plus award fee contract would make the most sense. The Armed Services Pricing Manual (ASPM) provides an excellent discussion of how to properly balance incentives.

2.4.6 Evaluation Criteria

Evaluation Criteria are used in competitive solicitations to assess the relative merits of competing proposals and provide the Government an objective basis for contract award. Samples of evaluation criteria for reuse are found in Section 7.1.

Evaluation Criteria for reuse should flow from the technical, schedule, management and cost program requirements. They should address the program's Most Important Reuse Requirements and Risks (MIRRRs). For example, if it is critical for a contractor to have excellent configuration management practices to assure software developed to be reusable is properly controlled, documented and updated, then this is a MIRRR and should be evaluated because it will produce discriminators among proposals. Conversely, if configuration management is necessary, but not critical, it is very unlikely any discriminator(s) will occur since any competent contractor should be able to meet the Government's minimum requirement here (the level at which the Government must write standards for evaluation). Figure 2-7, MIRRRs Produce Discriminators, describes the relationship of evaluation criteria to MIRRRs and contractor approaches.

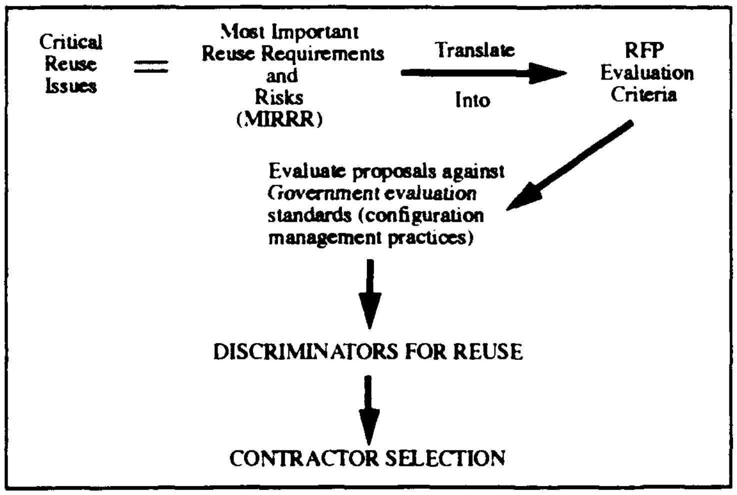

Critical Reuse Issues
=
Most Important Reuse Requirements and Risks (MIRRR)
Translate Into
RFP Evaluation Criteria
Evaluate proposals against Government evaluation standards (configuration management practices)
DISCRIMINATORS FOR REUSE
CONTRACTOR SELECTION

Figure 2-7 MIRRRs Produce Discriminators

Section 6.1 provides more examples of what might be typically considered important to evaluate when reuse is being practiced. Section 6.3 then provides examples of evaluation standards which relate to the evaluation criteria and help to assess offeror compliance with the Government's reuse requirements.

2.5 Acquisition and Legal Considerations

Introduction

Today's successful implementation of reuse is hampered by several impediments. The more significant among them are:

- Acquisition: There is a notable lack of policy guidance, training and tools for reuse strategies. This is now being tempered by availability of the DoD Software Reuse Vision and Strategy document, service reuse strategy documents and handbooks such as this Acquisition Handbook for program, engineering, financial, contracting and legal personnel.

- Rights: DoD policy (as stated in DFARS 27.4) currently exists which positions the Government against industry regarding ownership of software. Current DoD regulatory coverage (DoD Federal Acquisition Regulation Supplement (DFARS)) requires obtaining unlimited rights (or close to unlimited in the case of mixed funding) to most software components acquired under Government contract. The only reasonable exception (and some agencies will argue this case) to this policy is components which exist at the time of contract award. This policy discourages industry from investing in new concepts and developing new technologies where its competitive standing and/or commercial position can be instantly eroded by unnecessary Government rights positions. The Government's claims in rights apply to the delivered software component(s). Thus, a contractor is likely to use low-level technology and/or not engineer an easily upgradable software product knowing there is no economic incentive in the future. If the Government could be more adaptable and creative in negotiating rights, both the Government's and industry's interests could be protected through techniques such as escrow of rights, negotiated periods of contractor rights exclusivity, incentives and other means. Further compounding the problem is that it is not clear which Government agency accepts responsibility for reusable software components, with the interest of reuse, upgrade and standardization in mind. Typically, software components are delivered and users then operate and maintain their systems with no strong sense of making the software component(s) available to others. If the Domain Management function (as proposed in Section 2.1, Reuse Acquisition Planning Structure) were institutionalized, this situation could improve. Copyright and patent law regarding software is evolving and further complicates reuse strategies. The recent addition of DFARS 2 for competitive acquisition of commercial software has finally recognized commercial protections. However, this is only applicable to competitive acquisitions. The Government program manager's challenge is to balance everyone's interests and promote reuse.

- Regulations: Contracting, program and legal personnel are today forced to use a DFARS which is not well written, is confusing and often perplexing when dealing with software and data associated with software. There is little in the way of training or tools to aid the practitioner.

- Related to the software life cycle cost perspective is the fact that software is rarely static or useful by itself. User needs evolve over time. Technology advances the level of services that can be economically offered to users over time. Users need training and often request technical support. User manuals and tutorials are expensive to develop and usually require experience with the product over time to be refined and become more effective. Owning rights to software is owning a static snapshot of an evolving entity. It is owning only one part of a complete solution system. The full cost of providing the remainder of the total solution is then borne by the owner, since the developing contractor cannot amortize such costs over multiple Government and commercial users.

We offer these introductory points to stimulate your thinking when assessing reuse potential, making decisions regarding necessary software rights levels and creating your reuse acquisition strategy.

- The "Government" does not exist when it comes to software rights ownership responsibility. Specifically, there is no central agency chartered with managing and controlling software that the "Government" owns. This was not a problem when software was typically built to suit some unique agency need - that agency acted as "the Government" for purposes of determining software rights (no one else was interested anyway). The situation changes when we are dealing with reusable software components. Since there is no central "owner", typically the funding agency considers itself the owner and potential reusing agencies become customers. However, there is no established mechanism for one agency to "buy" the software from another, nor is there much incentive to "give" the software away, since there is always the possibility of bad press or requests for technical support and training. In addition, the agency's mission is usually only indirectly related to the software in question (especially if it is reusable by another agency!) and typically does not include providing technical support, installation and training services, updates, etc., for all possible Government "reusers" of such software, nor is there any incentive or justification for "advertising" the software to potential reusers across the Government. Yet this is exactly what must happen if the Government is to elicit any benefit from software reuse.

- Study after study has shown that software life cycle support costs are usually much larger than development costs, typically reaching 70-80% of overall costs. By claiming software rights, the Government automatically agrees to pay most, if not all, of these costs, since the contractor has little, if any, incentive to assume any of these costs since it cannot sell the software to any other Government agency. Coupled with the point above, this means that most software controlled by the Government has a much smaller user base and a much higher unit cost for support and maintenance than an equivalent package that can be sold commercially. Counter-intuitive as it may be, owning the software rights usually ends up costing the Government more than paying for its development and then giving it away to the developer (the same point applies to commercial firms who contract for custom-built software).

Although software component development for reuse or reuse itself will be supported by the normal acquisition process, there are legal issues which have a very significant, direct impact on reuse. These issues must be carefully considered in planning and implementing reuse strategies. To assure those strategies are successful, the following sections discuss these legal issues in detail, including steps to be taken and/or considered in your planning. Any potential for statutory or regulatory change will be noted. Revisions to this guidance will be provided as statutory or regulatory changes are made.

2.5.1 Software Rights

The right(s) to use software components must be clearly stated in any contract requiring software reuse. To understand the implications of this concept, a current perspective of statutes and regulations regarding ownership rights is necessary. Tables 2-10 and -11 [4], briefly describe what the DFARS includes in its definition of software, what levels of rights are possible, and its position regarding copyrights. These tables do not address patent rights. Simply stated, if the Government pays for the development of software (i.e., software is developed exclusively with Government funds), it retains unlimited rights to its use. If a contractor pays (i.e., software is developed exclusively at private expense) for development of software, the Government can only acquire restricted rights, as described in the table, unless it negotiates for some greater rights. In either case, a contractor would generally retain its copyright interest. For comparison, coverage at the FAR level is also included in Tables 2-10 and -11. As you can see, it is not quite the same regarding definitions of software, restricted rights or copyrights. DoD personnel should understand both FAR and DFARS coverage since industry must deal with both and often confuses DoD policies (DFARS) with other Federal Agency Policies (FAR). The DoD has determined FAR coverage is not suitable for its needs and has been allowed to create its own. It is also worth noting the DoD-Industry Advisory Committee on Technical Data Rights (Section 807 of the Fiscal Year 92 Defense Authorization Act) which is rewriting coverage on data and software for the DFARS. As of March 1994, a draft of the rewrite was not yet available; published reports indicate the revised coverage will improve industry's position on retaining software rights.

Table 2-11 Software and Ownership Definitions (Part A)

	Software Definition	***Government Unlimited Rights Software***	***Government Restricted Rights Software***	**Copyrights (Legal Right to Reproduce, Publish & Sell)**
DFARS	Computer software and computer databases (227.471)	Use, duplicate, release, disclose in whole or in part in any manner, for any purpose. Same rights can be given other parties (227.471)	• Use with the computer for or with which it was acquired (including locations where computer may be transferred) • Use on backup computer if primary com puter fails • Copy for safekeeping (archive) • Modify or combine with other software, assuring the derivative software based on restricted rights software contains the same restrictions • Any other rights not inconsistent with the stated minimum rights (227.471) • Commercial software restricted rights also include: title/ ownership remains w/contractor limit use to facility where computer is located limit use to facility where computer is located can not be made available to 3rd party without contractor's permission (252.227-7013 (c)(1)(i))	• Contractor authorized to copyright unless work (software) is considered a special work (227.476/252.227-7020), in which case work (software) becomes sole property of Gov't and is treated the same as unlimited rights software. Gov't granted nonexclu sive, paid-up license to reproduce, to dis tribute to the public, to perform or display publicly, and to prepare derivative works & have others do so for Gov't purposes (227.480/ 252.227-7013(e)) Note: There is disagreement today regarding whether software can be considered a special work.

	Software Definition	Government Unlimited Rights Software	Government Restricted Rights Software	Copyrights (Legal Right to Reproduce, Publish & Sell)
FAR	Computer programs, computer databases and documentation thereof (27.401)	Use, disclose, reproduce, prepare derivative works, distribute to public, perform & display publicly, in any manner, for any purpose. Same rights can be given to other parties (27.401)	Developed at private expense and is consid ered secret; commercial or financial and confi dential or privileged; published copyright software; including minor modifications of such software (27.401)	• Contractor authorized to establish copyright claim. Gov't granted paid-up nonexclusive, irrevocable worldwide license to repro duce, prepare derivative works, perform and display publicly by or on behalf of Gov't. Gov't license does not include right to dis tribute software to public(27.404(f)(iv)) • FAR subparagraph (g)(3)(i) under the Alternate III to the Rights in Data Clause 52.227- 14 provides the same restricted rights as DFARS with addition of providing software with same restrictions to support services contractors.

	Software Definition	Government Unlimited Rights Software	Government Restricted Rights Software	Copyrights (Legal Right to Reproduce, Publish & Sell)
Comment	DFARS treats computer software documentation as technical data, not software.	FAR provides specific treatment of rights for derivative works based on unlimited rights software.	DFARS definition is based on rights in use; FAR relies on basis of funding and/or control/ ownership of software.	• DFARS copyright license includes right to distribute to public where FAR license does not, unless special works is used.

Table 2-12 Software and Ownership Definitions (Part B)

	Commercial Software	**Government Purpose License Rights (GPLR)**	**Unpublished Software**	**Required for Performance Of A Government Contract or Subcontract**
DFARS	Computer software used regularly for non - Government purpose & is sold, licensed or leased in significant quanti ties to the general public at established, catalog or market prices.	Right to use, duplicate or disclose data (and software only in the SBIR program) in whole or in part, in any manner for Gov't purposes. Gov't purposes include competitive procurement but not commercial purposes. Gov't can authorize others to use for Gov't purposes.	Not yet released to public or furnished to others without restriction on further use or disclosure.	The development was called for in the contract, or subcontract, or it was accomplished during and was necessary for performance of a Gov't contract or sub contract
FAR	No formal definition. However, FAR 27.405(b)(2) references "existing computer software" as privately developed software normally vended commercially under a license or lease agreement restricting its use, disclosure or reproduction.	No similar definition.	None	No similar FAR coverage (note: FAR 52.227-14(b)(i) may be similarly interpreted)
Comment		GPLR would apply to any computer soft ware documentation for which the Gov't obtained such rights	DFARS definition encompasses devel oped software not yet, or perhaps never intended to be commercialized	Under DFARS 252.227-7013, Rights in Technical Data and Computer Software, subparagraph (c)(2)(ii), this language requires unlimited rights pass to the Gov emment. A contentious issue between Government and industry

DoD personnel should remember computer software documentation is considered technical data and has data rights, not software rights coverage (DFARS 227.401(6)), unlike the FAR which includes software related documentation in its definition of software.

Gray areas exist with respect to use of existing NDI software (not yet available in the commercial marketplace) or contractor-funded software which is developed at least partly in parallel with contract performance. DoD's policy (DFARS 27.4) is that the matter is settled with the Government obtaining unlimited rights. Industry often disagrees and will argue it has successfully obtained different levels of rights. Table 2-12, Alternative Approaches to Rights in Software, provides general guidance on rights issues to be considered for reuse when contractor software developed exclusively at private expense is involved. Remember, in this environment a contractor is likely to seek maximum rights retention to protect its investment. Balancing the Governments reasonable needs with the contractors interests should result in an acceptable agreement. The Government could even consider no more than an option for the appropriate level of ownership rights to be exercised within a stated period if there is no immediate need for Government exercise of its rights (See Section 7.7 for examples). This might be possible where the Government has not yet decided whether to support the software in-house, use a third party, or have the original developer maintain it.

Table 2-13 Alternative Approaches to Rights in Software

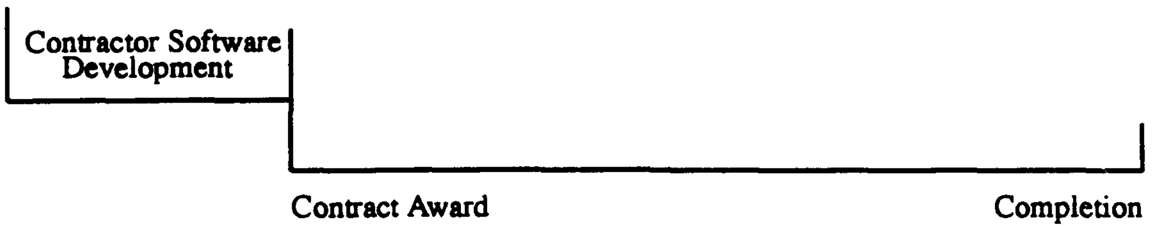
Contractor software developed and funded exclusively at private expense and used in performance of a contract	Government Rights Desired for the Software Component(s)		
Alternatives	Unlimited	Unlimited on the pro gram or some set of pro grams	Restricted
	Negotiate delivery of full rights OR Negotiate full or "program full" as an option to be exercised not earlier than X years after delivery of the software component(s) Negotiate full or "program full" as an option to be exercised not earlier than X years after delivery of the software component(s)	Negotiate license rights applicable only to program or to a set of programs	Already provided for in DFARS

When a contractor proposes to use (or reuse) software which it has independently developed, at private expense, or proposes to develop software independently, but in parallel with contract performance, current DFARS coverage (252.227-7013) states that if it is required in contract performance (defined in DFARS 227.401 (16)), the Government has unlimited rights regardless of where the funding comes from. This is a concept which focuses on "use" rather than "who funded development" to determine Government rights. Although this is the current regulatory position, industry typically will argue from a different perspective. Typical scenarios with industry for this concept are discussed below:

1. In Example A, Figure 2-8, the contractor should be able to easily argue that the software was developed independently and prior to award; thus, the Government should receive no more than restricted rights. However, the Government will argue that since the software is "Required for Performance" (DFARS 252.227-7013), the Government has unlimited software rights. There is little in the way of precedents (Armed Services Board of Contract Appeals or Court of Claims) to provide any definitive position on which party is "more right". Perhaps the Government could consider not seeking unlimited rights in order to: (1) Recognize the contractor's initiative and investment; (2) encourage reuse of the existing software components; and (3) foster additional industry incentives to invest and create reusable software. If greater rights are desired, alternatives such as those shown in Table 2-10 could be pursued in lieu of a time consuming and far less than certain approach of seeking unlimited rights by demand.

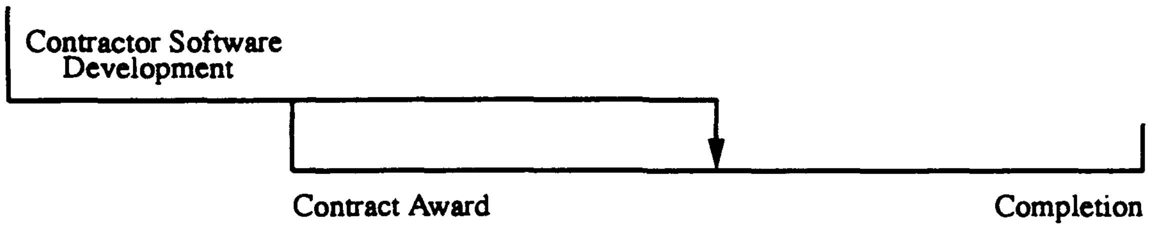
2. Example B, Figure 2-9, is somewhat more persuasive with respect to the Government arguing that the software is really being completed to enable its use in contract performance. Thus, the software is "Required for Performance" and the Government has unlimited rights. However, what if the contractor can show that substantial work was completed prior to award, following a business plan for product development, with a non-DoD (or even non-federal) market identified? Again, no clear Armed Services Board of Contract Appeals nor Court of Claims precedent exists. Typically, rights in these cases are either negotiated (See Table 2-10 above) or determined in the formal disputes or claims forums - with varied success for the Government and/or industry.

3. Example C, Figure 2-10, would typically present the strongest Government case for unlimited rights. Issues similar to those raised for Examples A and B could nonetheless be raised by the contractor and the resolutions would necessarily flow from negotiation, formal contract disputes or court decisions.



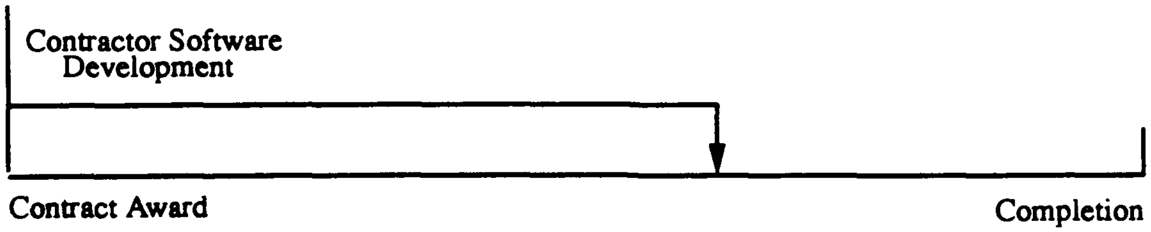
Contractor Funded Software Development - Completed Prior to Contract Award

Figure 2-8 Example A



Contractor Funded Software Development - Initiated Prior to Contract Award But Completed Subsequent to Award

Figure 2-9 Example B



Contractor Funded Software Development - Accomplished Parallel to Contract Award

Figure 2-10 Example C

Impacts on Reuse

Examples A, B and C have all affected the Government's desire to implement reuse strategies. When contractor funded or controlled software is involved, the issues raised are typically more troublesome than need be. Certainly none of these issues should preclude successful strategies for reuse when the Government has adequately considered its reuse objectives and planned for implementation. The following section will explore some new examples, coupled with copy right issues.

DoD's policy for copyright is also stated in DFARS 27.4, specifically 227.403-76. It allows the developing contractor to retain software copyright of any software product developed or generated under a Government contract, with a license granted to the Government. There are exceptions for what are known as "Special Works"; but, although not explicitly stated, software is not considered a "Special Work" (Note: DFARS 227.405-76 does not include software in its "Special Work" coverage). Thus, a contractor retains copyright, which affords the contractor the protection defined in DFARS 227.403-76 (a). Essentially, no one can commercially reproduce or use the copyrighted software unless permission is granted by the copyright owner. The Government's license allows it to use the software, or have others use it on behalf of the Government, for Government purposes only (e.g., for another Government contract). Contractors are somewhat concerned about this license, since it does allow the Government to distribute copies to the public, even though the public may not legitimately use it for other than Government purposes. As one might expect, industry believes that once the product is publicly disseminated, its ability to identify and pursue copyright infringement is significantly hampered.

The impacts of software copyright and previously described software rights on reuse must be understood. The following scenarios describe the most common situations typically encountered:

Scenario 1 Contractor A develops reusable software for the Government. Contractor A retains copyright, but the Government has unlimited software rights which essentially includes a Government purpose copyright license.

The Government provides the software to company B to reuse in another program. Company B cannot infringe on company A's copyright for commercial applications, but can take its "added value" product and claim copyright. Company A might, however, claim copyright of company B's product (the case law is unclear).

The Government should secure Company A's agreement in the RFP, and at the very start of its contract, that it recognizes the Government's intention to reuse the software and that it will not claim any copyright of derivative works products. In this way, Company B clearly understands its limitations and opportunities, and can make clear business decisions. 1992 New York Federal Court Decisions suggest contractor A cannot claim commercialized copyright for derivatives in any case (see below).

Scenario 2 Contractor A owns existing software (which is not yet available in the commercial marketplace) that it is offering to reuse/use in a Government contract. First, the Government must settle the question of whether use constitutes "Required For Performance", and if it does, whether it will try to argue for unlimited software rights. There is no simple answer here as previously discussed - either side may prevail in negotiations or litigation of the "Required For Performance" issue, if it is left to post award resolution.

The Government's RFP should:

1. Advise industry of the Government's absolute minimum needs regarding control of software rights for other than COTS products

2. Identify the Government's interpretation of "Required For Performance" regarding existing software

3. Require categorization of all software as COTS, existing, modified or new development

4. Advise of any alternatives regarding rights which are not acceptable

Industry can then reasonably assess the worth of its existing software versus the value of securing the program (and any related business). The Government can make evaluation judgements and negotiate what is acceptable prior to award. Post-award negotiations are typically drawn out and/ or antagonistic, which creates a poor business environment.

Copyright issues are clear in this instance. Contractor A is the owner. The same copyright considerations apply as in scenario 1 if the Government secures sufficient rights to allow it to pass the software product to another company. If sufficient rights cannot be obtained, copyright concerns become something of a moot point.

Scenario 2 is the extension of Example A, previously discussed above. It remains the most contentious case regarding software ownership and the extent of third parties' abilities to use that software. Where development of reusable software is a primary objective, the Government should consider either: (1) precluding use of existing software unless adequate and reasonable license rights can be negotiated; or, (2) establishing transfer of full, unlimited rights. In both cases, the issue of copyright must be settled to allow creation of derivative works with no further copyright claims by the original copyright owner. Section 6.2.6 contains a provision which attempts to identify similar issues.

Scenario 3 Contractor A initiates and/or completes parallel development of software it is offering to reuse/use in a Government contract. This scenario encompasses examples B and C, previously discussed. Here, a contractor proposes to complete its internally-funded software development while performing on a Government contract where the contractor's software will be used. Now the Government is faced with the added complexity of assessing whether the contractor has claimed any legitimate restricted rights, and how those impact reuse strategy. The rights (and copyrights) status must be identified and settled prior to contract award, if developing reusable components is a program objective. Delaying rights determinations and ownership/license issues until post award is certain to at least complicate, if not, doom to failure any effective reuse strategy. Note in earlier discussions (Section 2.5.1, Paragraph 3), the handbook cites the Government position in the DFARS that the Government has unlimited rights since the software was "required in performance" of the Government contract.

Scenario 4 Contractor A proposes to reuse existing, but not commercialized, software developed by company B. The reuse can be to satisfy an individual program requirement, or to integrate the existing component into a component being developed for reuse. The Government must understand the implications in terms of the ultimate program objectives. In addition to the RFP questions identified in Scenario 2, the Government must clearly establish:

1. The source(s) of funding for company B's original development and whether the Government already has some level of rights

2. Copyright status and what is necessary to allow derivative works to be created, (i.e., copyright license).

Current Copyright Status

The issue of copyright ownership with respect to software and what it entails is continually evolving. What was "gospel" six months ago may no longer be true today. Therefore, it is vital for the Government team to proactively look at copyright implications on reuse as part of its initial acquisition strategy. Legal counsel must be involved. NASA requires a pre-RFP assessment of copyright implications - this handbook encourages the same.

Some recent developments:

1. In early 1992, Apple lost its suit against Microsoft and Hewlett-Packard. Apple had claimed copyright infringement on the "look and feel" of its windows environment. The court found this not to be subject to copyright protection.

2. In June, 1992, a federal appeals court in Manhattan ruled that programs which incorporate the structure of existing software do not, in many cases, violate copyrights (Altair, Inc. vs. CAI, Inc.). The ruling is initially being interpreted to mean that little more than source code can receive copyright protection. If this ruling becomes a precedent outside the New York jurisdiction, the copyright issue for reuse will become significantly less complicated.

3. A July 31, 1992 court ruling on a Lotus vs. Borland copyright infringement favored Lotus, finding that Borland illegally copied part of Lotus's 1-2-3 program. Some feel that this contradicts the ruling in item 2 above - even though the judge cited that case and concluded that even under its tougher standard, Borland had infringed on Lotus copyrights.

Industry will want to preserve whatever level of commercial copyright protection it can. To the extent that this is possible, industry's incentive to develop and/or improve reusable software will increase. The Government can help this by also considering its right to "Duplicate and Publicly Disclose". Typically, there is little reason to disseminate information beyond Government programs' needs. Note that any language which addresses copyrights must conform to DFARS 227.403-77 (a)(4), which precludes any agreement prohibiting the Government from infringing copyright or patent. In the end, the Government is required to preserve this right, subject to reasonable compensation - this could be spelled out in an agreement on disclosure.

The Government may also want to explore whether copyright assignment (essentially giving the copyright to another party) should be a preferred option. For itself, this would require the military service secretary approval, if pursued.

Patents

The practice of patenting software is becoming widespread. While software patents have existed for some time, they have proliferated in the last decade. Many people argue that software is not and should not be patentable. Nonetheless, software is being patented, and the Government's ability to successfully practice reuse could be impacted if it does not adequately plan for the impacts of patents.

There are some significant differences between patents and copyrights. A copyright lasts for 75 years and is simple to claim - in fact, the notion of implied copyright exists for any original work whether a copyright is affixed to it or not. The existence of a copyright is immediately recognized when the copyright notice is prominently affixed to a product.

Patent exclusivity lasts for 17 years. Patents can only be granted by a Government agency, the U.S. Patent Office. The fact that someone has filed for a patent is not disclosed, nor is the nature of the invention publicly known until the patent is granted. So, the dilemma becomes, how do I deal with patents in software reuse? First, when developing reusable software and/or assessing whether to reuse software, the Government should:

1. Assess the practicality of performing a patent search to determine whether any patents exist which impact the development or the component to be reused. This may not be practical in terms of time or expense; or,

2. Ask the developer or the existing software originator to disclose existence of any patents and/or licensing agreements with other suppliers (multiple sources for the same software); and,

3. Have the contractor certify the software doesn't infringe a patent.

This takes care of existing patents. Now, what about patents applied for, but not yet granted?

> If a developer or supplier has an outstanding patent application, the Government can request disclosure, and use FAR 27 subpart 3 procedures to assure that it will be licensed if a patent is granted. This will allow others to be licensed on the Government's behalf (either at cost or on a royalty fee basis).

You may also consider whether to limit use of software to that for which appropriate and reasonable licenses have been, or can be established. Similar to copyrights, DFARS 227.403.77(a)(4) provides for Government ability to infringe on a patent. However, disclosure and agreement prior to award remain infinitely preferable to negotiation of licensing and compensation in a post-award, and perhaps non-competitive, environment.

Any license negotiated must be comprehensive enough to address all Government needs in development, operation, and post-deployment support, including modification and update/ modification.

During development of reusable software, your contractor may create software components that violate a patent ultimately granted to another developer. Since there is no practical way today to know if a patent application has been filed by someone not involved in your program, you will not know that a problem exists until and unless the patent is granted and the patent owner initiates a reimbursement claim. There are initiatives to require disclosure of patent applications 18-24 months after filing, but no one can predict whether these will come to fruition.

In the meantime, the "Phantom Patent" is best left out of the realm of planning for developing or reusing software components. FAR and DFARS coverage on patent infringements and protection should be adequate for reuse.

Current Patent Status
There is wide and significant disagreement in the legal and academic communities regarding whether software should be patentable. Today, software is patentable. Government acquisition managers should understand the potential impact, which is actually only significant when: a patent exists or may exist; it is an area vital to your reuse plans; and you are not aware of it (and thus cannot effectively deal with it).

To indicate the significance of software patents, in June 1992, Microsoft agreed to a sizable lump sum payment to IBM with continuing royalties on some products due to patent violation. IBM disclosed the existence of more than 1000 patents for software, addressing procedures as basic as the way a cursor moves (Source: Wall Street Journal, June 1992). Microsoft felt there were too many patents at such a basic level to ever effectively argue that it had not infringed in any of its software products. The lesson learned for the Government acquisition manager is that "patent disclosure would probably have sped up the settlement", or, "if known earlier enough, sent Microsoft down a different development path". In early 1994 Microsoft was found guilty in Federal Court of patent infringement on another company's software and was estimated to be liable for infringement damages in excess of $100 Million.

The League for Programming Freedom, which opposes software patents, is an excellent source for current status of software patents, alleged infringements and negotiated settlements.

Sections 6.6 and 6.7 provide examples for use when dealing with copyright and patent issues.

2.5.2 Liabilities

People within and outside the Government immediately raise the liability question when software reuse is discussed. When the Government provides software/software components, what responsibility does it assume? What warranty does it offer regarding product performance? What is the Government's liability if the furnished products fail when used as is, or if a derived software product fails?

If the original software producer has warranted the product, the Government may be able to pass that warranty on, unless its application is restricted to the original contract under which the software was produced. However, the warranty, at best, is likely to only cover the software in a specific application, and remedies are not likely to offer more than repair or replacement of the software, so that its intended operation is unimpaired. Reusing the software to create a derivative work would typically void any existing warranty, even if it could be passed to another contractor.

Therefore, if the software is used as is, a second contractor might have an opportunity to seek assistance under the warranty, although this is not likely. When used in a derived work, any warranty is probably voided. Can the Government then indemnify a contractor when it is provided with Government software? We would suggest that the question need not even be asked. A better/more correct question is "whether the Government should offer protection" and if it doesn't, is this a disincentive to reuse?

The Government should not attempt to offer liability protection when making software components available for reuse. It is not a practical alternative, and would require a prohibitively expensive testing and administrative organization. Rather, the Government should provide as complete a description as possible of the software (including source code if rights permit), all available documentation (including service reports) and identification of the original producer. Whether reuse is voluntary or mandated by the Government, the new contractor would have sufficient information available to make an intelligent technical and business decision regarding its ability to reuse the product and confidence in its quality. When reuse is mandated, the parties involved can construct added contract language to protect specific interests if contention arises during performance concerning each party's liability should contract requirements not be met. Section 6 offers examples addressing these scenarios.

Note that commercial software is used (reused) over and over without the liability issue ever surfacing. The significant proliferation of these products actually creates greater business risks than the limited-use instances for Government software.

Is the liability issue a disincentive? It certainly is in those instances where software is unproven, is provided with little or no documentation, evidences no records of update or service, or is furnished with such restrictive licenses that it becomes impractical to consider the product for reuse. One can contend that these circumstances would negate the viability of this software for reuse from the start, so the liability issue would become superfluous. Critics will argue that this position ignores the real world, where overzealous or inexperienced Government organizations will force reuse even in this environment. While rare instances of this nature could occur, a prudent contractor can and should refuse to participate under these circumstances. In any event, it is impractical to attempt to protect against this "exception" to the rule, beyond a Government commitment to adequately train acquisition personnel.

The Government must also recognize the need to provide a capability to ensure the independent testing/examination of software products made available for reuse. Alternatively, the Government could include a separate contract line item requiring the contractor to inspect the software products and report all flaws which could impact reuse.

2.5.3 License Agreements

License agreements simply spell out the executing parties' rights and obligations regarding the subject of the license. When dealing with copyrighted or patented software, you want to assure that the license provides sufficient current and future use coverage to accomplish all of your objectives. Table 2-14 below identifies some simple considerations in examining license agreements. Sections 3 and 6 of this handbook address what information to request in structuring license agreements and what a sample agreement would look like. DFARS 252.227 also includes examples of agreements. Another type of licensing agreement is known as a Cooperative Research and Development Agreement (CRDA). In a CRDA, the government helps transition its technology to interested parties but provides no financial assistance. Section 6 has a CRDA example and references for how they are used. CRDAs may be especially useful for someone willing to commercialize existing government software at their own expense. The CRDA would have to provide sufficient justification to the investor to make this worth while.

Table 2-14 Considerations for Reuse License Agreements

- Will the reuse occur once with no updates required (e.g., for a one-time, specialized test purpose)?
  A simple, one-time license should suffice
- Will reuse require continual modification and/or improvement across
  A single program
  Multiple, specific programs
  Unlimited applications
- Is the license perpetual or does it cease after a specified date or event?
- Does license expiration void continuing use for the same or future, similar applications?
- Does the Government want the option to unilaterally extend the license terms?
  Or, to extend application of the license to a broader universe?
- Does the Government want to escrow (protect) the software covered under license to allow:
  Assignment ot copyright to the Government if the owner abandons the product
  Ability to assure full rights to the software, including rights to use for any purpose (with or without fee) if the product is abandoned
- Does the license allow the Government to provide the software to other parties for Government purposes (competition, modification, maintenance, and integration with other products)?
- Is the royalty (See Section 2.5.5, Royalties) consistent with the nature and duration of the license?
  Frequency of royalty: paid-up; running; set period royalty
- If third-party software is involved, does the Government have sufficient rights/access to enable it to fulfill its requirements?

The license agreement should encompass no more that what is minimally needed. When addressing reuse, determination of license agreement requirements is more difficult since the reusable component may have broader application than you realize. Coordination with similar domain programs, PEOs, DACs, or similar officials is encouraged to assure that the widest possible reuse benefits are achieved. Similarly, license coverage must address all aspects of your program from development and throughout the program life cycle. A matrix similar to Table 2-15 could be used to assess needs for license coverage, and should be used in conjunction with Table 2-14 above.

Table 2-15 Anticipated Reuse Environment Matrix

Frequency of application	**Anticipated Reuse Environments**				
	Development	Test	Maintenance and Support	Modification	Derivatives
Single Program Application					
Multiple, specific program applications					
Unlimited, Government program applications					
One-time use					
Multiple, but limited, use					
Unlimited use					
Other					

The challenge in structuring/negotiating an adequate license is that while your individual program needs may be met, the Government's broader needs may go unfulfilled. Domain management, PEO/DAC and Office of the Secretary of Defense (OSD) level organizations should be consulted to coordinate positions on:

1. Ultimate responsibility, over time, for maintaining, updating, supporting the reusable component(s)

2. Funding source(s) for (1) and any royalties due

3. Extent of license application

   - Site-specific

   - Federal Government wide

   - Term limited/unlimited

2.5.4 Royalties

Royalties are simply payments to a party for use of its property, similar to paying rent on a house or apartment. The royalty is associated with a license (Section 2.5.4), and is typically:

1. Paid-up royalty. This is typically a lump-sum amount, paid at one time.

2. Running royalty. This would be a set, recurring amount paid over the term of the license. For example, if a patent had 5 years remaining, the royalty would be paid over those 5 years whenever the product/patent was used.

3. Set period royalty. Similar to a running royalty, but limited to, for example, less than the full patent protection period; or, when a certain dollar amount was reached or a set number of uses was documented. This is a potentially more finite warranty than described in 2) above.

4. Variations on all of the above can be structured.

When addressing royalties for software to be reused, Table 2-15, Anticipated Reuse Environment Matrix, should be used to estimate the extent of reuse and then determine the most appropriate royalty type. From the Government's perspective, the royalty amount should be reasonable and in the context of:

1. Anticipated total savings to the Government over all anticipated reuse occurrences.

   - Cost avoidance (Development, maintenance, training)

2. Reasonable rate of return (profitability) to the licensor

   - Commercial potential/actual use should be included to insure the Government does not absorb a disproportionate share of the royalty.

3. Any existing Government or commercial warranties.

   - Note, the Government can demand the lowest royalty rate (DFARS 252.227-7002, Readjustment of payments)

Note that the Government should never pay any royalties for a copyrighted work if it has unlimited rights or GPLR. Similarly, the Government has paid-up rights to patents which were inventions under Government contracts (FAR 27.302(c)).

An example of use of the Reuse Environment Matrix and determination of appropriate royalty follows.

Table 2-16, Example Anticipated Reuse Environment Matrix, describes a situation where a number of programs will have specific reuse applications, some of which will result in derivative products. The exact number of occurrences should be finite, but is not exactly determinable. Since multiple programs are involved, each should bear its share of the royalty and have separate agreements (if all programs were in the domain of a PEO, a single agreement could be considered).

Table 2-16 Example Reuse Environment Matrix

Frequency of application	Anticipated Reuse Environments				
	Development	Test	Maintenance and Support	Modification	Derivatives
Single Program Application					
Multiple, specific program applications	XX		XX		XX
Unlimited, Government program applications					
One-time use					
Multiple, but limited, use	XX		XX		XX
Unlimited use					
Other					

A paid-up royalty may not be feasible, given the unknowns (risk to the Government of overpaying). Either running or set period royalties might make the most sense - the nature of the ownership characteristics (e.g., patent, duration, likelihood of technology stability) would create a third dimension to finally assess which is most appropriate. If paid-up royalties are pursued, the Government should assess their value using a formula similar to:

$$\sum_{(i=1)}^{n} (A_i \times I_i) = R^p, \text{ where:}$$

A_i = Number of reasonably anticipated applications

I_i = Unique investment for each application

R^p = Total paid-up royalty

The royalty amount would then be estimated using the factors described above. The Government would then have a credible basis (standard) for evaluating and negotiating a reasonable royalty payment, as well as assessing the economic utility of reusing the particular software component.

Sections 3 and 6, respectively, discuss what information should be included in an RFP and what clauses might be used in a contract.

Note: This template is also useful in identifying the inputs to cost effectiveness assessments by identifying the anticipated frequency and types of reuse.

2.5.5 Incentives

There is a wide variety of contract incentives in existence today which may be applied to achieve objectives. Some incentives are direct financial (e.g., royalties discussed in 2.5.5; or cost incentives and award fees discussed in FAR Part 16). Others are more indirect, such as a cash incentive to not exercise the Government's legitimate unlimited rights for reusable software components until some period of time has passed. Disincentives, such as a non-cash incentive to recoupment and Government demand for unlimited rights and/or assignment of copyright have been previously discussed.

Table 2-17, Incentives in Reuse Environments, discusses various incentives and their characteristics in reuse environments. It is important to note that non-cash incentives may be should be used for both Government and commercial markets.

Table 2-17 Incentives in Reuse Environments

Reuse Environ-ment	**Incentive Type**					
	Award Fee	Cost Incen-tive	Royalty	Delayed Gov-ernment Rights Clain (Planned)	Dererred Rights Claim (Unplanned)	Restricted Government-Copyright License
Develop reusable software com-ponent(s) Government funds	• More sub-jective • Easier to ad-dress critical reuse devel-opment criteria (quality, documenta-tion, etc.) • Contractor at risk	• More difficult to con-struct • Easier to measure only for objective criteria	• Promotes reuse of patented or copyrighted software	• Contactor opportu-nity to obtain Govern ment and commer-cial advantage	• Some op-porunity for contrac-tor to obtain Govern-ment and commercial advantage	• Contractor commercial and/or Govern-ment advantage

Encourage reuse of existing software	• Same as above	• Same as above	• Encourages both creation and reuse of patented or copyrighted software	• Same as above for derivative works product unique to developer	• Same as above for derivative works product unique to developer	• Same as above for derivative works pro
Privately fund creation of reusable software	N/A	N/A	• Opportunity to recover development costs and obtain reasonable profit	• Increases contractor incentive to create reusable software when Government claim can be validated.	• Some incentive for contractor to create reusable software when Government claim can be validated	• Contractor commercial and/or Government advantage
Share funding for reusable software creation	• Increased contractor motivation • More subjective • Easier to address critical reuse development criteria • Contractor risk	• Inreased contractormotivation • More difficult to construct • Easier to measure only for objective criteria	• Promotes both creation and reuse of patented or copyrighted software	• Contactor opportunity to obtain Government and commercial advantage	• Some opporunity for contractor to obtain Government and commercial advantage	• Contractor commercial and/or Government advantage

A matrix similar to Table 2-18, Government/Contractor Reuse Objectives, should be used to understand Government and contractor objectives and to identify non-conflicting intersections. These intersections will help identify the most appropriate incentives for reuse. The matrix should be used for both Government and commercial markets.

Table 2-18 Government Contractor Reuse Objectives

CONTRACTOR OBJECTIVES	GOVERNMENT OBJECTIVES				
		Single Program Reuse	Multiple/ Specific Programs Reuse	Unlimited Programs Reuse	Public Disclosure
	Near-Term Investment Recovery				
	Long-Term Investment Recovery				
	Near-Term Competitive Advantage				
	Long-Term Competitive Advantage				
	Elimination of Competition				

Table 2-18, Government/Contractor Reuse Objectives, should be used to establish potential compatibilities/incompatibilities for each market.

Monetary motivators such as award fees, cost incentives and royalty payments are typically effective mechanisms when properly structured. Each is discussed below. Examples are provided in Section 7.

Award Fees

These are discussed in FAR 16.305 and the corresponding DFARS. Development will require very high quality domain analysis, software engineering, test and documentation. These characteristics are typically better assessed in a subjective environment which an award fee can accommodate Similarly, when attempting to (1) identify reusable software components, (2) improve/modify existing software to make it reusable, and/or (3) actually reuse existing components, award fee criteria can be structured to place emphasis on the unique characteristics of each of these environments. As reuse programs mature/change, award fee criteria can be modified to reflect the current environment.

The flexibility of award fees makes them especially attractive. Industry generally reacts in a positive manner to these fees, although the lack of "guarantee" tends to create some level of apprehension, until the contractor believes that the Government consistently and fairly applied the criteria.

Consideration should be given to a separate award fee. If the basic contract is Cost Plus Incentive Fee (CPIF), adding an award fee solely for reuse makes the contract structure CPIF/Award Fee and focuses on reuse as an added incentive, instead of one which covers all aspects of contract performance.

Cost Incentives/Royalties

FAR 16.304 and 16.401 discuss cost incentives. Royalties are covered in FAR 31 and section 2.5.5 of this handbook. Each of these is a more finite incentive than an award fee. Both require very definite criteria to assure objective measurement/assessment for payment.

A cost incentive may be useful/appropriate when attempting to motivate a specific degree (percentage) of reuse on a program. For example, the Government will pay a certain, fixed amount for achieving a stated percentage of reuse. As you can see, this requires more front-end Government work to determine what percentage is realistic, how it can be effectively measured, and what conditions (timely delivery, documentation acceptability) will be imposed to assure that the reusable software is satisfactory in and of itself and within the overall software system.

Royalties are most useful in encouraging reuse of existing components and in encouraging industry-funded development of reusable components. The Government's willingness to compensate a contractor for the cost of reusing another's software product or for its investment in creating a reusable product will:

1. motivate more analysis to identify a broader range of available software components,

2. create more products,

3. create a commercial environment in which industry will be motivated to continually improve its software products.

The Government must plan for and fund royalties. This should not be a problem, since program costs would be expected to be lower overall when reusing products, even with royalty payments included. Nonetheless, the Government must assess the reasonableness of the royalty (see section 2.5.5). The Government must also assure that the royalty license provides sufficient flexibility to enable creation, maintenance and/or improvement of software products (see Section 2.5.1, Ownership Rights). The Government must also consider costs of administering royalty, such as tracking, invoicing and payment.

Deferred Delivery and Deferred Ordering

DFARS 227.405-71 addresses both deferred delivery and deferred ordering.

Simply stated, deferring delivery means that the software is generated and documented under and in accordance with the contract, but the actual delivery of software documentation is deferred. A contractor may well be motivated to produce more reasonable software if, through deferred delivery, it understands that the Government will create some period of time during which the contractor can enjoy both a Government and commercial advantage. Clearly, the Government must have the minimum software documentation in time to fulfill its program needs (such as

Users and Operations Manuals), but alternatives exist - such as providing for contractor software maintenance until all the documentation is available. This is a somewhat radical concept, but it's one that should be considered in the proper environment. The government benefits by the original developer maintaining the software; the contractor benefits through protection of its competitive position.

Deferred ordering means that the software documentation has not been but can, at some point, be ordered and delivered under the contract. While at first this appears to be a better motivator than deferred delivery, it is not. In this instance, a contractor is more subject to the Government's whims. The contractor may create a more reusable component (perhaps even partially investing to help commercialize it), only to find that the Government decides to order it much earlier than either party ever envisioned, or never orders it at all. It is still potentially effective, but would be even more so if the Government agreed not to exercise its rights prior to a certain date or milestone event.

Restricted Copyright License

As discussed in section 2.5.1, the Government obtains specific license rights under a copyright. It is feasible to consider altering some of those rights (e.g., public dissemination or withholding rights for a reuse project to create derivative works) to increase contractor motivation to create reusable software knowing its competitive position is protected. These alterations may require a deviation to the DFARS requirements, but should be considered when the potential technology payoffs are significant.

2.5.6 Warranties

Subsection 2.5.2, Liabilities, suggests that software warranties are often not practical in a reuse environment. This does not mean that they cannot or should not be pursued. It merely points out that software warranties, like other warranties, cannot be substituted for effective planning, development, maintenance and quality documentation of software components.

FAR 46.7 and DFARS 246.7 discuss warranties fairly extensively, but in terms of "systems", "hardware" and "technical data". Of course, the warranty clauses identified in the regulations (such as FAR 52.246-19, "Warranty of Systems and Equipment under Performance Specifications or Design Criteria"; DFARS 252.246-7001, "Warranty of Data" can be tailored to more explicitly address software. However, one must first decide:

1. What is being protected with a warranty?

2. The desired/required remedies.

3. The duration of the warranty.

4. The cost-effectiveness of the warranty (see DFARS 246.770-8).

Table 2-18, Analysis Tool for Assessing Warranty Effectiveness/Applicability, can be used to help assess whether a warranty might make sense. Sometimes you will have no choice, at least at the system level. During the 1980s, 10 U.S.C 2403 statutory coverage was added, requiring warranties for "weapon systems" (see DFARS 246.703 and 246.770-8). Warranties for weapon systems, however, can be tailored and/or waived. Table 2-18 can help in determining whether warranty coverage for reusable software components should be included and is applicable to both contractor-supplied and Government-furnished software.

Table 2-19 Analysis Tool for Assessing Warranty Effectiveness/Applicability

	Reusable Software Approach		
	Software component devel oped to be reusable, or mod ified for reuse	Existing software component identified as reusable "as is"	Commercial software com-ponent
Software component charac-teristic requiring warranty consideration	• Reusable assets should perform to the level de-scribed in supporting documentation Developers of reusable components should support this concept	• Same as for new and/or modified software, but: Original developer may not be willing to war-rant for reuse Any existing warranty may be exclusive of or voided by reuse else where	• Typically, only the stan-dard commercial warranty will be offered May be sufficient Commercial vendor may consider extended coverage
Remedies required	• Fix the software compo-nent to perform at documented levels • Consequential damages possible if reuser adequately tested com-ponent prior to reuse, but defect not detectable. Consider use of liqui-dated damages	• Same as for new and/or modified software, but: conditions above still apply	• Consequential damages typically excluded • Performance to docu mented levels warranted

Warranty duration	• Some reasonable period after delivery - typically not more than 1 to 2 years Software may not change, but prudent business person would not commit to longer periods	• Same as for new and/or modified software	• Commercial limits 90 days to 1 year typical
Cost/benefit analysis results	• What is the added warranty cost over the warranty period? • What is the likelihood that the component will be reused in the warranty period? • What is the likelihood that a failure can be discreetly identified? • What are the administration costs?	• Same as for new and/or modified software, but: Also assess whether reuse warranty costs necessarily duplicate any existing warranty costs	• Included in COTS price • Usually difficult to analyze added costs for extra coverage Commercial pricing protection

You certainly would want to:

1. Have any commercial warranties passed on, giving you the option to later decide whether they are useful.

2. Warrant no proprietary/patented components are used - if that is a requirement (See Sections 2.5.1 and 2.5.4 for discussion of these issues). It may only be practical to warrant if the reuse component developer has and knowingly used any of its protected software (unless it gives full Government rights) or anyone else's.

Remember, any warranty should offer added protection (insurance) beyond the time the software is accepted. If there is a real need for that protection, a reasonable warranty can be written. Its cost effectiveness is a separate issue. Sections 3 and 7 provide more information.

3 REQUEST FOR PROPOSAL (RFP)

3.1 Proposal Instructions

Proposal instructions should be worded in such a way that the offerer is not told what the response should be, but is allowed to be frank. A candid response will allow the Government team to make an objective determination of whether the respondent will be the appropriate candidate for the task at hand.

Information required from respondents is to be included in the management, technical and cost sections of the proposal. The management proposal section should include information regarding training, organizational structures and management initiatives. The technical proposal section should include information concerning expertise and technical approach. The cost section of the proposal should reflect the monetary impacts of the technical and management approaches.

3.1.1 Management Proposal

The contractor's/offerer's management proposal should address several issues which can impact reuse success. These issues can include: organization and experience, risk management, personnel, and program execution/reviews and controls. Each of these is briefly discussed. The focus is placed on information which should be requested in RFP Proposal instructions. This will assure that sufficient discriminating material is available to enable assessment of the best reuse approach. Specific examples are provided in Section 6.

Organization and Experience
In evaluating a proposal, you will want to know whether or not reuse impacts the offerer's organization, and why. We would expect the offerer's systems and software engineering organizations to at least specifically recognize the need for focused reuse efforts in areas such as requirements analysis and functional allocations. An offerer may include in its proposal specific management reporting on reuse tasks to highlight their importance. Typical questions to be answered include:

1. What is the impact, if any, of the software reuse tasks on the offerer's organizations?
    - e.g., is creation of a domain expert function required?
2. Where impacts are identified, provide supporting rationale, describing anticipated program benefits.
    - e.g., standardization, improved processes
3. How does the offerer integrate software reuse into its systems and software engineering tasks?

4. What is the offerer's current experience with software reuse, and how is it applicable to the current reuse task? What types of continuous improvement practices are employed to advance the practice of reuse?

5. If the offerer has no reuse experience, what techniques (training, teaming, etc.) will be used to provide a sufficient reuse knowledge base to successfully execute program tasks?

6. What employee incentives/motivations/training opportunities exist to promote reuse as a methodology in the offerer's organization?

7. How will the offerer's systems and software engineering methodology integrate reuse?

   - If reusable components are being developed, the offerer should be tasked to provide information on: how it will perform domain analysis; whether prototyping or similar techniques will be used to validate architecture decisions; use of CASE tools, generation of documentation; and similar issues.

   - If components are being modified for reuse, the offerer should be tasked to provide information similar to the above and how the offerer will assure the functional and physical integrity of the components to be modified, including an assessment of the suitability of existing support documentation.

   - If COTS/GOTS are being used, the offerer should be tasked to provide information on its validation process to (1) verify the COTS/GOTS capabilities and (2) their suitability for the intended reuse application

8. How do the offerers standards and practices incorporate reuse activities into the design, development, coding, testing and integration functions of its software process?

9. What concepts/activities can the offerer identify (as methods and/or actual practice) on how reuse components have impacted maintenance and support?

Risk Management

No new risk management methodologies should be required for software reuse projects. Any viable risk management methodology should be capable of integrating software reuse tasks. What is important to note are any peculiar risks which may be associated with reuse. All typical software development risks apply. Table 3-1, Risk Issues Associated With Reuse, below, identifies some considerations to be addressed in the proposal instructions.

Table 3-1 Risk Issues Associated with Reuse

	Reusable Software Approach		
	Software component developed to be reused, or modified for reuse	Existing software component identified as reusable "as is"	Commercial software component
Potential Risk Areas	• Lack of organization structure to support reuse Technical and manage ment • Insufficient numbers and categories of personnel • Lack of expertise • Inadequate documentation	• Same • Same • Same • Same	• Same • Same • Failure to understand need to assess COTS documentation • Viability of the COTS supplier and proba bility of continuing maintenance and update availability

Personnel

There is an acknowledged shortage of software engineers in the United States. No near-term solution is in sight. The pool of available systems and software engineers who understand and appreciate reuse constitutes a small critical mass. Today's educational institutions and development methodologies emphasize new development. There are only a few, innovative efforts to systematically integrate reuse into the software engineering process. We have a cultural bias against reusing something in preference to developing anew. We have a cultural history of planned obsolescence and throw-away products, which is a natural inhibitor to development of reusable software components. Some critical questions to be answered in any reuse project include:

1. Given the numbers and categories of systems and software engineering personnel proposed, how many have reuse experience versus the actual number required? What kind of reuse experience do they have in creating reusable components or reusing existing components (given the immaturity of the field)? How will the offerer fill the shortage, if any, with sufficient trained/experienced personnel in time to support program schedules?

2. What training is available to personnel in the engineering discipline (internal/external)? What training will be provided? What is the source? How has the offerer assessed and maintained its quality? How many of the offerer's existing proposed personnel have successfully completed the training? What is the offerer's capability (class limits, academic schedules) to support program requirements? What are the qualifications/credentials of the instructors?

3. Is reuse proficiency considered in personnel performance evaluations? If so, is it sufficient to create a cultural change/awareness which positively contributes to the program's potential for success?

4. If specific educational requirements are stated, do the proposed personnel meet these?

Program Execution/Reviews and Controls

Successful program execution depends on, among other critical factors, sound planning, thorough reviews and use of management controls which allow proactive rather than reactive approaches to emerging problem areas which could impact a successful reuse effort. Again, our focus is on those issues peculiar to reuse - all programs require sound execution and control. Table 3-2, Management Indicators and Techniques for Reuse, identifies some of the more critical aspects to be identified in a proposal to enable the government to assess the offerer's reuse capabilities.

Table 3-2 Management Indicators and Techniques for Reuse

	Program Execution	Program Reviews	Program Controls
Management Indicators and Techniques for Reuse	• What techniques are used to validate reuse approaches? • Do "workarounds" exist for all reuse approaches categorized as other than low risk? • Do software development plans incorporate and ade quately address reuse? • Are test criteria adequate to assure quality reusable components and supporting documentation?	• Is reuse specifically included in program reviews? • How do informal reviews integrate reuse considerations? • Do reviews address types and use of metrics for (See Section 2.4.3 for references to metrics) progress improvement?	• What metrics are proposed for reuse? Have they been used before? What is their success record? • Do cost budgets reflect sufficient allocations for develop ment of reusable software? recognize savings where software is reused? • Do organizational reporting structures encourage or suppress reuse initiatives?

3.1.2 Technical Proposal

The instructions for the technical proposal section of the RFP can combine both the expertise and approaches requested from the offerer. A respondent's expertise in software reuse can be evaluated by examining their past experiences and knowledge of the various aspects of soft ware reuse. Of course the particular proposal instructions will depend upon the technical and reuse goals and objectives of a particular program. (See Section 7.2, Proposal Instructions, for specific examples and see Section 4.3 for evaluating the technical approach.) An offerer's technical approach to software reuse can be separated into: domain requirements planning, reusing existing software components, developing reusable components, library development, sources of existing components, component evaluation, and risk minimization approach. Software reuse processes must be integrated into and be compatible with the software development processes. Software *reuse development tools* and *environments* will also effect the successful incorporation of existing components and the development of reusable components. Both experience in reuse and risk management plans for each of these areas should also be requested from the offerer.

If the program's reuse objectives include reusing existing software components, the processes, techniques and methods proposed must be explained in full. How will reuse impact the soft ware development processes used? Where will the contractor obtain components? Do they plan to prototype system requirements using existing components? What prototyping plans and processes will be used? What methods will be used to integrate the software component(s) into the system design? How will the reliability or maintainability of the system be impacted?

When Government provided software components are made available to offerers, additional specific issues should also be considered, such as: Does the offerer plan to leverage Government provided common architectures, software, hardware and COTS?; Have potential Government reuse sources and candidates been identified?; Is the offerer proposing the use of a layered architecture with adequate separation of concerns so that reusable components can be effectively "plugged in" during both development and post deployment?

Information related to technical expertise that should be provided by respondents includes: domain requirements planning, software development (both developing reusable components and reusing existing components), library related, knowledge of software reuse environments/ tools, and specific domain expertise.

3.1.3 Schedules

Program schedules and impacts of reuse were discussed in Section 2.5.2. An offerer's proposal must provide detailed program schedules which incorporate and clearly identify specific reuse activities and milestones. Among the significant activities and milestones requiring visibility in a reuse proposal are:

- Requirements and domain analysis completion
- Requirements and domain analysis completion Identification of reuse alternatives (functional allocations)

  Unique development

  Modify existing components

Reuse Fxisting components "as is"

Develop to be reusable

- Milestones for:

  Prototypes, simulations

  Design review activities

  Test

  Delivery and validation of documentation

- Workarounds for other than low (or just "high") risk areas

3.1.4 Cost Instructions

The cost portion of the Instructions For Proposal Preparation (IFPP) should contain specific language which conveys to the contractor how the Government wishes to see software reuse cost data portrayed in the proposal. The IFPP should contain a description of what the Government considers to be adequate cost visibility to allow a proper evaluation of how costs were estimated. Costs by task should be asked for, and the tasks must be defined. If the contractor is reusing existing components, his cost proposal should separately show costs for domain analysis and modeling, architecture development (although these two activities are not likely to be conducted on individual system acquisitions), acquisition of reusable components, and testing and integrating reusable components, so that the Government can verify the legitimacy of proposed costs in each of these task areas. For all newly-developed software, whether developed for reuse or for one-time use, the traditional separation of development costs by phase (e.g., requirements analysis, design, code, test and integration) will apply. If the contractor uses a model-based approach to develop its cost estimates, model inputs, data bases and/or other information substantiating the inputs, and outputs should be requested as part of its proposal submission.

To support Government evaluation of the contractor's proposed costs, the Government will need to know and, consequently, should request information on the following: software size (which must be clearly defined, e.g., source lines of code or function points) for both new and reused code; the amount of integration of reused code with new code; the software personnel quality (in terms of their experience with the language, application, development tools, modern programming practices, and reuse tasks); and the development environment (in terms of the availability of development and support tools, turnaround time, and other relevant environmental factors). These are data elements that are normally requested to facilitate the over all evaluation of the contractor's proposed software development costs and to determine its validity.

Refer to Section 7.2 for specific language to be included in the IFPP.

3.1.5 Royalties

Royalties are discussed in section 2.5.5. Offerers should be tasked to identify the following in their proposals:

- Any software and related documentation subject to royalty payments
- Why use of this "protected" software is consistent with program requirements and does not limit all Government envisioned future reuse environments
- Why royalties are more cost effective than outright purchase of the software rights
- Why other alternatives (i.e., use of other software, development of new software) are not viable or as technically or economically attractive
- Protection (warranties?) provided as part of the royalty payment

The Anticipated Reuse Environment Matrix, Table 2-15 in section 2.5.5, Royalties, can also be used to develop additional proposal requirements, focusing on the type and frequency of the known and anticipated reuse application(s).

3.1.6 Rights

Section 2.5.1 discussed the issues of Government versus contractor software rights, copy rights and patents. It also pointed out some of the difficulties (e.g., pending patents) which prevent certainty in identifying whether restrictions exist in use of software. Nonetheless, it is a relatively straightforward process to request identification of rights issues which will impact reuse. Some of the questions that should be posed to offerers in the RFP are:

- Does the software offered have any limitations with respect to Government use for:

  Any Government purpose?

  Any purpose?

- What is the nature of the limitation?

  Restricted Rights?

  Copyright?

  Patent?

- Is the offerer willing to grant a license to the Government for:

  Use on this contract?

Use on other Government contracts?

Creation of derivative works?

For Government purposes

By the Government or other parties

Dissemination to others for creation of derivative works for commercial use (e.g., either initially through a Government contract or directly disseminated to the public to stimulate creation of commercial derivative works)

Dissemination to other contractors for maintenance

- Would the offerer assign copyright of any newly developed software (including modification) or existing software?
- What is the cost of any license granted to the Government?

  Demonstrate its economic utility

- Are there any other architectural alternatives available which involve the use of "protected" software? If so, why and what?

While some answers to these questions may be difficult to deal with, the questions are easy to ask. Typically, rights only become a "real problem" when the questions are not posed, or resolution of the issues raised by the answers is not settled until after contract award. Current Government policy on software rights is sufficiently controversial that these issues must be resolved prior to contract award for any successful reuse program. An example of an Army program's decisions and how they were implemented in RFP Section L instructions is found in Section 7.2.6.

3.2 Government-Supplied Information

Government information chosen to be supplied in the Request for Proposal (RFP) is dependent upon the acquisition and reuse goals for the particular system that is being acquired. Reuse-related information (including the Software Reuse Strategy) should be provided, so that each contractor's plans and proposals are developed from the same baseline. In addition, it is usually more efficient and more cost effective for the Government to research its own pro grams for reuse-related information, rather than have contractors do so.

The information to be included in an RFP can be grouped into the following categories: reuse data (domain knowledge, technology base, Government projects), sources for components, descriptions of components, and software reuse guidelines and standards. Some of this information, such as domain analysis results and locating component sources, may actually be tasking for the contract at hand rather than information supplied in the RFP. On the other hand, some of the topics, such as certification criteria, development standards/guidelines and domain models, are not currently standardized by the Government, but may be in the future. However, some guidelines and proposed standards do currently exist in various Government organizations, and are being used on a limited basis. A general discussion follows on each of these topics. References for additional information are found at the end of each category.

Reuse Data

Data that should be supplied to offerers consist of domain knowledge, the general reuse technology base of software tools, models and methods, and other applicable Government projects. The results of previous domain analysis studies should be provided. If a domain analysis is to be performed for the particular acquisition, the contractors should be able to use the approach that they either have used or are familiar with. They should be able to use any reasonable approach that adequately meets the scope of the domain in question. The respondent's proposed domain analysis approach should be described in detail in the proposal. Information about related reuse projects as well as Government points of contact for a particular application domain area should also be included. Examples of this data follows:

Domain Knowledge

- Domain definition: characterization of the functional boundaries of a domain (what is and is not included in the particular domain)
- Representation of the primary inputs, outputs, and interfaces of the software within the application area
- Domain glossary: standard definitions and terminology *of the domain objects* and their interrelationships
- End users' perspective of capabilities of applications within domain
- Generic software requirements and other specifications: descriptions of the operating environments, operating policies, algorithms, and potential hardware platforms
- Generic architecture design specification: the high-level packaging structure of functions and data, interfaces and control to support the applications in a domain

Technology Base

- List of available software reuse tools, models and methods

Government Projects

- Short description of related software reuse projects, how they fit into current acquisition and how they are expected to fit into the current application
- Points of contact for domain information (if applicable)

References

- Criteria for Comparing Domain Analysis Approaches, Draft, Steven Wartik, Ruben Prieto-Diaz, Software Productivity Consortium, 1991
- Domain Analysis Bibliography, Software Engineering Institute, CMU/SEI-90-SR-3, June 1990
- Domain Analysis Process, Interim Report - Domain Analysis Project, Software Productivity Consortium, DOMAIN_ANALYSIS-90001-N, Version 01.00.03, Jan 90

Component Sources

To properly implement software reuse, offerers need to be assisted in locating and possibly evaluating software components. Thus, sources of reusable components must be supplied in the RFP. In addition, respondents should be encouraged to identify any additional libraries or components (commercial, university or in-house) which they propose to consider for use. The following information should be supplied to offerers in the RFP:

Applicable Government reuse libraries (include contact information)

Applicable COTS/GOTS products

Description of how further detailed information of these products will be made available

Products which must be evaluated for use in the software effort

References

Reusable Software Components, Dr. Trudy Levine, Fairleigh Dickinson University, STSC Cross Talk, March 1992 (listing of reuse sources)

Component Description

If components will be provided during the effort, the following descriptive information about those components should be provided in the RFP.

How component fits into domain

Source language of component

Description of component (what it does)

Authors

Ownership of software (copyrights, patents and data rights)

Legal and security restrictions

Liabilities (if any)

Warranties (if any)

Derivative work restrictions (if any)

Certification level

This information will allow the respondents to make intelligent decisions in their development plan, based on any impacts, both positive and negative, due to using the components.

Guidelines and Standards

For each type of component, two types of guidelines can be provided: (1) general engineering guidelines applied during the development of components, and (2) standards and criteria for accepting both existing components and newly developed components for a particular system. A program or library may have varying levels of acceptance criteria, depending on the application and goals and objectives of the system. Examples follow:

General Criteria

Applicability - relevant to application domain

Completeness - contains all references and required items

Consistency - complies with established standards

Maintainability - allows modification with minimal impact

Portability - platform independence

Reliability - low error rate

Testability- includes test plans (passing some criteria)

Understandability - appropriate documentation

Types of Components to be Evaluated

Domain model

Software architecture

Product design

Implementation components (code; test plans, procedures and results; and software/system documentation)

References

Reuse Library Process Model, Software Technology for Adaptable, Reliable Systems (STARS) Program, IBM Federal Sector Division, 26 Jul 91

Reuse Metrics and Measurement Concept, Draft, Joint Integrated Avionics Working Group (JIAWG), Prepared by (JIAWG) Software Task Group, 28 Sep 90

Software Reuse Guidelines, U.S. Army Information Systems Engineering Command (USAISEC), U.S. Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), Apr 90

Software Reuse Handbook, (Annotated Outline), Reifer Consultants, Inc., Joint Integrated Avionics Working Group (JIAWG) Reusable Software Program, 27 Sep 91

Volume 1, Fundamental Concepts, Appendix C, Process Model Descriptions

Volume 2, Management Concepts

Volume 3, Technical Guidelines

Volume 4, Library Guidelines

Volume 5, Reuse Standards

A Software Reuse Maturity Model, STSC Conference, Apr 92, Phil Koltun, Anita Hudson, Harris Corporation

STARS Reusability Guidelines, Prepared by IBM for Electronic Systems Division, 30 Apr 90

3.3 Contract Data Requirements List

The RFP package should clearly delineate the types of data that the Government needs in order to monitor the reuse program and analyze its effectiveness. In some cases, tailoring existing data items can accomplish these objectives, but in others, unique data items will have to be employed, as described below.

3.3.1 Management

Existing management status reports can be used to capture the software sizing and personnel data that are relevant to the reuse scenario. Key to the understanding of the reuse program are data on the software size in total and the breakout of the code that is newly developed for reuse, newly developed for one-time application, reused (from another source) with modifications, reused as is, and COTS. These data elements should be required for each reporting period, and should be portrayed as planned and actual lines of code. With these data, changes in code size and allocation by category can be tracked.

Software personnel skill data should also be requested, specifically aimed at an understanding of the software reuse-related skills of the current staff, to include the domain engineer, application domain experts, and the remainder of the software engineering team. In addition, data which reflect the number of software personnel involved in software reuse tasks by period should be required. A simple format which asks for head count (by skill category) by reuse task (e.g., domain analysis, domain modeling, architecture development, component acquisition and validation, and component test) can be a valuable aid in determining whether the con tractor has applied the proper resources in the proper quantity to the effort.

3.3.2 Technical

Government-requested data on software reuse include several unique data items in the technical area.

The Software Reuse Plan should be an integral part of the Software Development Plan. It should identify and document both the reuse objectives, and the methods and criteria that will be used to create, acquire, modify, and maintain reusable software components. Additional information on the Software Reuse Plan is presented in Sections 4.3 and 6.2 of this document. The Software Library Report documents the choice of a domain-specific reuse library(ies) to be used for acquiring and depositing reusable components. Its contents should describe the program's library requirements, the applicable domain, the necessary hardware and software, the criteria that were used in evaluating libraries, the particular libraries that were evaluated, the results of the evaluation, and the rationale for selecting a particular library. Refer to Sections 4.3 and 7.2 for more information on the Software Library Report.

The Library Development and Management Plan is needed when a contractor is required to establish and maintain a domain-specific reuse library. It describes the technical requirements for the library, the domain engineering process and approaches to be used, the recommendations for operational hardware and software, and the classification schemes and mechanisms. In addition, it should discuss the contractor's management policies and procedures regarding the library hardware and software; domain modeling; acquisition, evaluation, testing and modification of components; configuration management; and user support. See Sections 4.3 and 7.2.1 for more information on the Library Development and Management Plan.

3.3.3 Schedule and Cost

Contract data items related to cost and schedule should be tailored to provide visibility into the impacts of software reuse on a program, in order to support both monitoring of current contract performance and collection of reuse metrics for subsequent use. Schedule reports should identify software development tasks at a low enough level of detail to support visibility into and management of both normal software development and reuse tasks. Consequently, schedule reports should include timelines for higher level reuse tasks (e.g., domain analysis, domain modeling, architecture design, component acquisition and validation, component test and integration), as well as for development tasks (e.g., requirements analysis, design, code, test and integration, and documentation). Cost reports, such as the Cost Performance Report (DI-F-6000) or the Cost/Schedule Status Report (DI-F-6010), should be tailored to require reporting at a low enough level to support monitoring of cost performance and variance analysis for the software tasks delineated above. Data collection at this level also allows construction of a reuse cost database to assist in cost estimation and evaluation of future reuse efforts.

4 EVALUATION AND SELECTION OF CONTRACTORS

4.1 Standards for Proposal Evaluation

Standards for Proposal Evaluation (Evaluation Standards) are measurement guides used by Government evaluators to assure that proposal responses comply with Government requirements. Standards are written at the minimum acceptance level, i.e., what is minimally necessary to comply with the stated requirement. They are written against the lowest level of evaluation. For example, shown below are typical evaluation criteria structures. The different terminologies merely represent service/agency designations.

Table 4-1 Proposal Evaluation Terminology

Typical Structure	Alternate Terminology
Area: Technical	Factor: Technical
Factor: Software Architecture	Subfactor: Software Architecture
Subfactor: Software Reuse	Element: Software Reuse

[Note: Evaluation Criteria are discussed in section 2.4.6]

A standard for evaluating reuse in proposals would be written at the subfactor (or Element) level. Thus, the detailed evaluation would be performed at that level, then summarized at the factor, then finally, the Area level. If no subfactors or Elements existed, the standard would be written at the Item (or Subfactor) level. Standards provide consistency and objectivity in the evaluation process.

Each service has regulations discussing standards and their use in proposal evaluations. Listed below are some of the higher-level regulations, by service, which discuss standards, and how they should be developed and subsequently used in the proposal evaluation process

Table 4-2 Proposal Evaluation Regulations

Air Force	Army	Army
AFARS Appendix AA	AMC-P715-3	NAVAIR
AFARS Appendix BB	Vols 1-4	NAVSEA 9OR/CIT

Evaluation standards are either quantitative or qualitative (or some combination) measurements. Evaluators essentially assign (no matter how complex the rating scheme) one of three ratings after evaluating the proposal against a standard:

Table 4-3 Proposal Evaluation Rating Definitions

Rating	Definition
• Acceptable	• Meets the standard
• Unacceptable	• Fails to meet the standard Proposal is deficient
• Exceptional	• Exceeds the standards in a way which provides useful benefits to the Government

When structuring a competitive program involving reuse, evaluation standards will be necessary to assure a consistent and objective evaluation process. Section 7.3 provides some sample standards as guidelines in structuring your reuse program.

4.2 Management Approach

Section 3.1.1 discussed what types of information would be typically requested when trying to assess whether a contractor has established a satisfactory reuse approach. If, for example, Organization and Experience was sufficiently critical to the reuse effort, and could be expected to produce discriminators among offerers, it would probably be identified as an evaluation criterion as follows:

Area:	Management
Item:	Organization and Experience
Factor:	Reuse

Again, the standard would be written at the Factor level. It would assess:

- Whether reuse has been properly integrated into the management process
- How the organization addresses reuse tasks
- What absolute minimum levels of personnel experience (and types of labor categories) the Government considers necessary for program success
- Other standards deemed necessary

Section 6.3 includes a specific example of this standard as an illustration.

The respondent's organizational structures and management initiatives needs to encourage software reuse in its organization and positively influence software personnel performing the particular development. Some things to look at include:

- Relationships with other contractors that encourage software reuse

- Incentive programs that reward employees for successful reuse initiatives
- Use of Total Quality Management principles
- Establishment of a corporate/organizational reuse library product lines and generic domain-specific architectures reflected in a firm's investments

The software reuse skills, expertise, and roles needed depend upon whether existing components are being reused, reusable components are being developed, domain analysis functions are being performed, or a reuse library is being developed or managed. The contractor's team should collectively have the needed skills and experience as specified below [12] for the particular type of effort:

- Performing a domain analysis
- Reusing existing components
- Developing reusable components
- Developing and managing a reuse library

No matter which type of skills are needed, the offerer should have experience in developing systems in the domain as well as knowledge of functions, operations, procedures, principles and current technologies of the documents at hand.

PERSONNEL ROLES:
DOMAIN ANALYSES AND MANAGING A DOMAIN MODEL [CECOM89]

Domain Analysis Technologist

Define languages, tools, techniques to be used in performing domain analysis

Train personnel in use of methods (if applicable)

Application Domain Expert

Expertise in application domain being analyzed

Direct experience building systems in that domain

Ideally has expertise in application domains related to domain of interest

Address commonalities across domains

Domain Analyst

Performs analysis and documents domain model

Domain Analysis System Specialist

Expertise in using specific domain analysis tools

Expertise in storage and retrieval of components from reuse libraries (This will be needed when the reuse process becomes more automated).

4.3 Technical

A respondent's technical approach and corresponding experience as they relate to software reuse must be evaluated. The following checklists of questions are ones that can be used in evaluating a proposal. The appropriateness and applicability, of course, depend upon the system being acquired.

Technical Expertise

DOMAIN REQUIREMENTS PLANNING

- Has the respondent performed domain analyses?
- What domain analysis approach(es) have they used?
- If more than one, which one do they prefer? Why?
- What domain modeling technique(s) have they used?
- If more than one, which one do they prefer? Why?
- In what domains or application areas were these analyses performed?

SOFTWARE DEVELOPMENT: DEVELOPING REUSABLE COMPONENTS/REUSING EXISTING COMPONENTS

- What software reuse process(es) have they used? How did it/they fit into the software development process used?
- What were some impacts on the reuse objectives resulting from the combination of the reuse and software development processes (positive and negative)?
- What components have been developed to be reusable (architectures, specifications, requirements, code, test suites, and documentation)?
- What guidelines, standards and metrics have been used?
- What impact did these have on prior projects?
- What prototyping techniques have been used?

LIBRARY

- What classification schemes have they used in library development or as a library user?

SOFTWARE ENGINEERING ENVIRONMENTS/REUSE TOOLS

- What software reuse environment(s) has the respondent used?
- Has the respondent developed any related software tools: library mechanisms, software development?
- What software reuse tools have they used?

DOMAIN EXPERTISE

- Into which domains or application areas does the respondent's software development experience fall?
- In what subdomains (user interfaces, communications, etc.) does the respondent have software development experience?
- What aspects of software development were performed for these domains (designing, developing software, documentation, IV&V, etc.)?
- Does the respondent have domain experience as a user?

Technical Approach

DOMAIN REQUIREMENTS PLANNING

- What domain analysis approach will be used?
- How will the domain knowledge be represented?
- What tools/environment will be used to capture domain knowledge (boundary, scope of domain)?
- Are these methods and procedures compatible with the requirements (technical, schedule, cost, risk)?
- Are the domain modeling techniques adequate for the scope of the domain in question?

REUSING EXISTING SOFTWARE COMPONENTS

- What method(s) will be used to integrate the software component(s) into the offerer's design?
- What impact will this have on the cost, schedule, and requirements of the system being developed?
- Will prototyping techniques/procedures be used, and if so, which methods will be used?
- Will prototypes reuse existing software?
- Will prototypes be developed for the target environment and language?
- How will the reliability and maintainability of the system be impacted?
- Does the offerer have the types of commercial software, hardware, and software tools needed to meet reuse requirements?
- What components have the respondent developed which are being proposed for reuse?

SOURCES OF EXISTING COMPONENTS

- Does the offerer specify sources of existing software?
- Does the offerer present products that are potentially reusable?
- Does the offerer provide candidate components for future inclusion into the effort?
- Have licensing/data rights been defined?

COMPONENT EVALUATION

- What special activities are needed to ensure the reusability of components?
- What techniques/methods/tools are to be used?
- What are the techniques and acceptance criteria that will be used to validate software components?
- Are these activities and techniques compatible with the requirements (technical, schedule, cost, risk)?

DEVELOPING REUSABLE COMPONENTS

- What software reuse processes/procedures will be used?

- What software development processes/methodologies will be used?
- How wi'l the reuse and software development processes be integrated?
- What Software Engineering Environment and tools will be used to create and test components?
- What software metrics and reuse development standards will be used?
- Does the offerer have the types of commercial software, hardware, and software tools needed to develop reusable components?

RISK

- What does the offerer identify as the potential benefits and risks of:

  Developing reusable software

  Reusing existing software (as is or modified)

  Using the proposed software reuse methodology/techniques

- How does the offerer identify risks associated with software reuse?
- What are the plans to mitigate those risks:

  Management (planning, personnel, resources)

  Technical (reliability, maintainability)

  Schedule

  Cost

- What are the estimated productivity and reliability gains or losses (for the reuse-based approach)?

LIBRARY DEVELOPMENT

- What methods are proposed to control and manage configuration of the reusable components?
- What is the proposed Software Engineering Environment and tools to be used in cataloguing and configuration control? How will they interact with the reuse library?
- What classification schemes will be used in organizing the library's contents?
- What library mechanisms/tools will be used?

- Are these plans compatible with the library requirements?

Documentation

This section discusses the purpose and contents of three software reuse-related documents (Software Reuse Plan, Software Library Report, and Library Development and Management Plan), since drafts of these may be required in the proposal. A Software Reuse Plan should be required, whether reusable components will be developed or existing components will be incorporated into a software development. A Software Library Report and Library Development and Management Plan apply to choosing and establishing a library, respectively. Since these documents are written during the period of performance, an evaluation discussion of each is provided in Section 5.2, which discusses post-award contractor performance.

Software Reuse Plan

The Software Reuse Plan must be based upon the Software Reuse Strategy developed by the Program Office and the Acquisition Strategy Panel. Thus, this information must be provided in the RFP.

The Software Reuse Plan should be required when existing components will be incorporated into a system development and when components will be developed to be reused. This document should identify and document the objectives, methods and criteria for creating, acquiring, modifying, and maintaining reusable software components. To assist in making documentation more reusable, the Software Reuse Plan should be an integral part of the Software Development Plan (SDP), regardless of whether it is a section of the SDP, an attachment or another volume. This decision would, of course be made by the Program Office. (See Section 7.2, Proposal Instructions. for contract wording. For information on evaluating the Software Reuse Plan, see Section 5.2.)

Software Library Report

If the development effort requires the contractor (rather than the Government) to choose a library, then a Software Library Report should be required. This report should document the choice of one or more domain-specific reuse libraries. First, the report should describe the program's requirements for a library: What domain should the library cater to? What hardware and software are necessary to access the library? What types of components are needed? Is the library based on a generic architecture? Based on these requirements, the report should include the criteria used in evaluating libraries and a list of those being evaluated. The report should contain the results of the evaluation, the library(ies) chosen, and the rationale for its selection. The effort may also require the contractor to establish and maintain a relationship/interface with the library in order to populate the library with reusable components and reuse existing components. (See Section 7.2, Proposal Instructions, for contract wording. For information on evaluating the Software Library Report, see Section 5.2.)

Library Development and Management Plan

When a contractor is required to establish and maintain a domain-specific (horizontal or vertical domains) reuse library, the development and maintenance plans should be a required document deliverable. The library development portion of the plan should describe the technical requirements for the library, the domain engineering processes and approaches that will be used, operational software and hardware recommended, and classification schemes and mechanisms. It should also address the library's ability to easily integrate into the software engineering environment. The library management portion of the document should discuss: management plans; policies and procedures regarding hardware and software; domain modeling; acquiring, evaluating, testing, and modifying components; configuration management; and user support. (See Section 7.2, Proposal Instructions, for contract wording. For information on evaluating the Library Development and Management Plan, see Section 5.2.)

4.4 Cost Evaluation

The Government's evaluation of proposed cost should be aimed at the realism inherent in the contractor's overall software development costs and, specifically, in his reuse costs. If the information described in Section 3.1.4 is requested, this evaluation is possible. The Government can compare information it has accumulated elsewhere on software reuse costs by task (e.g., domain analysis and modeling, architecture design, component acquisition and validation, component test and integration) with that supplied by the contractor in response to the IFPP. Areas of major difference will become immediately apparent, and can become discussion items during the evaluation process. Obviously, the Government's ability to conduct such comparisons will be enhanced by its own successful data collection, using tailored data items on other reuse contracts, as referred to in Section 3.3.3. The larger the Government database, the better will be the Government's ability to assess the contractor's proposed reuse costs.

For newly-developed software, the Government evaluation process will be much the same as it is now. Using the contractor's inputs for size (new vs. reused), personnel qualification, development environment, required integration, and the like, the Government evaluator can exercise an appropriate model to generate estimates of software development costs (see Section 6.4), which can be compared to the contractor's proposed costs to assess realism during the evaluation process. Of course, the evaluator will have to make the necessary adjustments to model inputs, as described in Section 7.4, if the software is being developed with the objective of subsequent reuse.

5 CONTRACT MANAGEMENT

Software reuse data should be reported to the Government. As in other software development efforts, development metrics are used by the Program Office to evaluate both the planning and progress of the software reuse effort. To properly measure the progress and effectiveness of a software reuse effort, metrics should be collected on management, technical, schedule and cost factors. Progress metrics should indicate any deviations between planned and actual milestones for both incorporating existing components and developing reusable components. As in other software development acquisitions, reporting data should be required to be delivered to the Program Office prior to any reviews, so that the data can be reviewed, analyzed and necessary questions and feedback formulated. This will assist in making productive use of both formal and informal Government reviews.

5.1 Management

Although necessary domain and other technical data are provided to the contractor in the RFP, additional information must also be provided to the contractor as the need arises. New methods, techniques, and tools used to refine the software approach 'are currently and will continue to emerge to effect the institutionalization of software reuse. The Program Office should continually monitor these current efforts and determine how they will affect the acquisition at hand. Consequently, if these efforts will impact the system under development, both the pertinent information and application details, should be provided to the contractor.

The primary management metrics pertaining to software reuse that should be examined are the software size and personnel metrics. To evaluate both the development of reusable components and the incorporation of existing components into development efforts, software size metrics should be collected and evaluated. For each reporting period, both planned and actual lines of code for the following should be reported: total, newly developed, reused as is, and reused/modified [14].

The contractor should also report the planned and actual staffing levels against software reuse skills and tasks. The primary software reuse skills are domain engineering and application domain expertise. The software reuse-related tasks that staffing levels should be reported for are: domain analysis, domain modeling, architecture development and component acquisition, modification, development, and test.

In addition to measuring both the progress and plans of the effort, the newly developed or modified components need to be evaluated to determine if they can be reused in other systems. Here is a checklist to assist in performing this evaluation [15]:

- Can the newly developed components be reused?
- Are the components sufficiently generic to meet other Government requirements?
- Do the components meet the quality level required?

- To what degree do the components exceed minimum requirements?
- What is the potential for cost savings (based on cost/benefit analysis)?
- How does the actual cost differ from the proposed cost?
- How did developing reusable components affect the overall schedule?
- How did developing reusable components affect productivity?

When existing components are reused in a software development effort, the following checklist can assist in evaluating the results:

- What percentage of the software system consists of reused components?
- What percentage was required vs. what percentage did the contractor propose?
- What was the cost savings resulting from reusing existing components?
- How does this compare with the proposed cost?
- How did reusing existing components affect the overall schedule?
- How did reusing existing components affect productivity?

The following list can assist in calculating productivity and cost savings [16]:

- Number of times the component is used
- Type of component
- Amount of change required for reuse
- Associated errors resulting from use

5.2 Technical

The guidelines and criteria for evaluating components can be similar, whether the components are being developed for further reuse, existing components are being reused or components are being submitted to a library. Effective reusability is influenced by the developer's experience, the methodology used during development, and the system's performance requirements.

Criteria used to evaluate components should be based on an established domain analysis and generic architecture, as well as sound software engineering and quality practices and procedures. Measuring the effectiveness, and reusability of components will be useless without an architecture baseline. Both the producer (Government or contractor) and consumer (using command/or acquiring command) need to have a common understanding of what is to be measured, how it will be measured and the significance of the results. Thus, criteria need to be tailored for each system, but can be based on the following development guidelines.

Each component type must be evaluated separately: domain model, software architecture, product design and implementation components (code, test plans, procedures and results, and system/ software documentation). The lower-level components are dependent upon the higher- level components. For example, an architecture should not be developed without a domain model, and code not written if there is no design.

There are general criteria that each type of component needs to be evaluated against. For example, both requirements and test descriptions should be evaluated in terms of their applicability to the domain. Specific requirements from a software requirements document must be traceable to generic domain requirements, and test descriptions must meet the general test requirements of the domain.

GENERAL CRITERIA

- Applicability - relevant to application domain
- Completeness - contains all references and required items
- Consistency - complies with established standards
- Maintainability - allows modification with minimal impact
- Portability - platform independent
- Understandability - contains appropriate documentation
- Functional capability
- Non-functional constraints (e.g., security features)
- Modularity - a change to one component has minimal impact on others
- Generality
- Self-descriptiveness
- Traceability

DOMAIN MODEL

- Traceable to and represents domain

- Contains information to facilitate domain architecture development

SOFTWARE ARCHITECTURE

- Allows applications within a domain to reuse high-level designs
- Need a complete domain model
- Need validation and verification with domain experts

PRODUCT DESIGN

- Need a complete domain model and generic architecture

IMPLEMENTATION COMPONENTS

- Testability- includes test plans (passing some criteria)
- Reliability - low error rate
- Meets general criteria

ADA CODING CRITERIA

- Frequency of static component references
- Similarity of coding style
- Similarity of subprogram invocation sequence
- Components hidden from external visibility
- Frequency of instantiating generic packages
- Degree of component independence
- Degree of component coupling
- Degree of component cohesion
- Similarity of component documentation
- Extent of type definition reuse
- Frequency of a component's being renamed
- Appearance of components in common directories

CERTIFICATION CRITERIA

- Is the component accompanied by formal specification and verification documentation?
- Does the documentation include other documentation to enable modification (e.g., PDL)?
- If code, does it follow any endorsed coding guidelines?
- Does the component conform to standards for reusability, complexity, portability?
- Is the component accompanied by a maintenance agreement?
- Is there documented evidence of successful, frequent reuse?
- Is the component guaranteed by some organization?
- Are there any disclaimers attached to the component?
- Is the component submitted with test plans, procedures, and results?
- Are the software development tools used in the development of components compatible with the required certification criteria?

Library

Development guidelines and evaluation criteria for components accepted for inclusion in a domain-specific library should be more generic than those for a particular system. Although the above certification criteria should apply to library contents, additional library-unique criteria may be required, such as, the ability of code to run on multiple hardware platforms. It is critical that certification criteria be established to maintain a consistent level of quality for components introduced to any reuse library.

Documentation

Planning documentation is usually required during the first phases of system development. Thus, evaluating required documentation is located here, since they are delivered and reviewed during the contract monitoring stage. Following is a checklist to use in evaluating the Software Reuse Plan, Software Library Report, and the Library Development and Management Plan.

SOFTWARE REUSE PLAN

- Is the Software Reuse Plan consistent with the program's Software Reuse Strategy?
- Is the Software Reuse Plan consistent with the system's requirements?
- What are the objectives of the plan?

- How does it fit into the plans for software development?
- What are the methods and criteria for creating, acquiring, modifying, and maintaining reusable software components?

SOFTWARE LIBRARY REPORT

- Example Library Requirements
    - Component types

        Domain Model

        Software Architecture Specification

        Product design

        Implementation Components

            Code

            Test plans, procedures, and results

            System/software documentation

            Test cases

            Input scenarios

    - Search mechanism

        Search for and extract components (comprehensive, easy to use)

        Browsing capability

        Able to adjust search time

- Classification scheme

    Domain-specific

- Metrics - able to store:

    Reusability data

    Quality data

    Configuration data

    Software complexity data

    Level of test

Program Trouble Reports

- Certification data - different kinds and levels of certification

  Quality

  Ease of reuse

  Degree of use

  Amount of reuse

- Representation capabilities

  Domain architectures

  Component relationships

  Capability to execute or demonstrate a component

  Capability to store typed components and invoke views appropriate to component type

  Ability to store and display non-textual components and information

- Capability to update a component and associated data

- Interfaces to external tools

- Interfaces to other repositories (interoperability, data transfer capabilities)

- Adherence to open system standards

LIBRARY DEVELOPMENT AND MANAGEMENT PLAN

Development

- Requirements

  Library technical requirements

  Audience's needs (knowledge)

  What is the domain(s)?

- Domain Model

  - How will the domain knowledge be captured?

    What processes/approaches will be used?

    Which domain analysis approach will be used?

Which domain modeling approach will be used?

- Operational software

  What classification scheme(s) will be used?

  What library mechanism(s) will be used?

  What database system(s) and method(s) are proposed?

- Operational Hardware

  Communications

  Distributed networks

  Storage size/approach (backup/archive)

  Interconnections

- Library software tools

  Modeling tools

  Prototyping

Management

- Hardware/Software

- Domain Knowledge

  Domain model

  Components

    Acquire

    Evaluate

    Test

    Modify

  Configuration management

- User Support

5.3 Schedule/Cost

Software reuse does not alter the fact that time is a critical metric. Schedule performance will be measured in terms of, for example, the time it takes to accomplish a domain analysis or to perform domain modeling, the time it takes to develop an architecture that takes full advantage of reuse opportunities, the time it takes to search out domain libraries to find and validate reusable components, and the like. Managers will also want to measure the reduction in schedule due to reuse of components (i.e., the elimination of design, code, and test phases for specific components). Newly-developed software will also be measured in terms of the time spent in each phase of development (analysis, design, code, test and integration, documentation).

Similarly, currently used metrics in the cost arena can be adapted for software reuse. Standard lines-of-code-per-day or cost-per-line metrics that apply to current software development also apply when developing software specifically for reuse. They may, however, reflect lower productivity (i.e., fewer lines-per-day or higher cost-per-line) when developing for reuse, as described in Section 2.4.3. Possible productivity reductions can be assessed once reuse cost data from a significant number of programs has been collected and analyzed. However, there may be savings during test and PDSS where reuse efficiencies are measured.

Of course, there must be some emphasis placed on devising cost and schedule metrics uniquely applicable to reuse scenarios. PEOs/Domain Managers should be exploring these metrics and determining how they will be developed.

6 SOFTWARE REUSE AND THE FEDERAL INFORMATION RESOURCES MANAGEMENT REGULATION (FIRMR)

As software reuse grows in acceptability and practice, we can assume more commonality in categories of software components. This commonality may ultimately cause some software components to become sufficiently generic that they no longer meet the exceptions of FIRMR 201-1.002-2, which today exempt most DoD equipments and software from FIRMR coverage.

In any software program, it is incumbent on the DoD manager to examine FIRMR 201,Part 4 and FIRMR Bulletin A-1 to assure themselves of compliance with all Federal Regulations (including and beyond the FAR and DFARS) which support the acquisition of software components. The FIRMR introduces no impediments, it merely recognizes the regulatory responsibilities of the General Services Administration in the acquisition of information resources.

FIRMR Bulletin C-12 describes the Federal Software Exchange Program (FSEP) and can be found in full text in Section 7.1.1 of this handbook. The FSEP is a library resource of common-use software, with supporting documentation, for which the government possesses full ownership rights. This library should be considered when establishing domain needs.

7 EXAMPLES

This section includes sample text, provisions and clauses for possible use. They should not be used verbatim. Each program has its own characteristics, some of which will suggest alternate ways to use reusable components. We recommend that those characteristics be identified by using the models and matrices in this handbook. Only then can the material in this section be usefully applied.

Where applicable, the source of the examples is identified if they are not original products of this handbook.

The examples are also included to stimulate your thinking. You will probably be able to either improve on them, or use them to develop your own unique text. Handbook sections 2 and 3 should be reviewed prior to considering use of any example from this section.

7.1 Evaluation Criteria (Section M)

7.1.1 Federal Software Exchange Program (FSEP)

FIRMR Bulletin C-12, 30 Jan 91

TO: Heads of Federal agencies

SUBJECT: Federal Software Exchange Program

1.Purpose. This bulletin provides information and guidance on the Federal Software Exchange Program (FSEP).

2. Expiration date. This bulletin contains information of a continuing nature and will remain in effect until canceled.

3.Contents.

Topic	Paragraph
Related material	4
Information and assistance	5
Definitions	6
Acronyms	7
Program description	8
Agency responsibilities	9
Identification of common-use software	9a

Software maintenance	9b
Data	9c
FSEC responsibilities	10
Cancellation	11

4. Related material.

FIRMR 201-21.403

5. Information and assistance. The Department of Commerce, National Technical Information Service (NTIS) publishes the "Directory of Computer Software" that may ordered from:

Department of Commerce

NTIS

5285 Port Royal Road

Springfield, VA 22161

Telephone: FTS or (704) 487-4650

Order Number: PB 88-190962

6. Definitions.
"Common-use software" means software that deals with applications common to many agencies, that would be useful to other agencies, and is written in such a way that minor variations in requirements can be accommodated without significant programming effort.

7. Acronyms.

FIP	Federal Information Processing
FSEC	Federal Software Exchange Center
FSEP	Federal Software Exchange Program
NTIS	National Technical Information Service
OTA	Office of Technical Assistance

8. Program description. The FSEP is applicable to common-use FIP software developed or revised by or for Federal agencies. It is not applicable to software that is classified or proprietary. It is not applicable to software to which the Government does not possess full rights of ownership. The FSEP is administered by the Department of Commerce's NTIS and GSA's Office of Technical Assistance (OTA). The program encourages the sharing of common-use FIP software and related documentation. Federal agencies report their common-use software to the FSEC. FSEC facilitates Government-wide sharing of reported common-use software.

9. Agency responsibilities.
a. Identification of common-use software. To identify and make available common-use FIP software, as required by FIRMR 201-21.403, Federal agencies should take the following steps:

1. Review software within the agency to identify common-use software that may be of use to other agencies.

2. Prepare and submit abstracts of identified common-use software to:

   Department of Commerce

   NTIS - Federal Software Exchange Center

   5285 Port Royal Road

   Springfield, VA 22161

3. Notify FSEC promptly of changes to or problems with common-use software previously reported.

4. Make a one-time submission, within 15 days of receipt of the FSEC request, of the common-use software and its supporting documentation. Sufficient documentation should be provided to facilitate implementation by other users. It should contain, as a minimum–

   (i) A narrative;

   (ii) User instructions, which should include program interface requirements; system resource requirements; identity of the computer on which the software is operational; program language; the name, number, and release of the system under which the software is operating; applicable data communications interface requirements; and applicable error message descriptions with recommended corrective actions;

   (iii) A logic flowchart to indicate the ease of removal or addition of program modules;

   (iv) Sample inputs and outputs; and

   (v) Program listing of the source and object coding as well as available cross-reference listing generated by the applicable assembler or compiler.

b. Software maintenance. The submitting agency is not responsible to another agency for maintenance of common-use software submitted to FSEC.

c. Data. No data files or data bases will be included with the common-use software. No private or personal data will constitute any portion of the common-use software and supporting documentation to be reported and exchanged by FSEC.

10. FSEC Responsibilities. FSEC's responsibilities include–

a. Maintaining a central library of summary descriptions of common-use software, including a complete index of the inventory and master copies of requested software and supporting documentation;

b. Providing a copy of the requested common-use software and supporting documentation, at the published price, to a requesting agency;

c. Editing, screening, and compiling agency abstracts of common-use software submitted for exchange;

d. Functioning as a central point of contact with agencies for information and dissemination of available common-use software;

e. Helping agencies identify common-use software to satisfy their requirements;

f. Helping agencies obtain information concerning technical problems with common-use software released through the FSEC by referring users to the source of the common-use software; and

g. Notifying agencies of changes to common-use software obtained through FSEC.

11. Cancellation. FIRMR Bulletin 32 is cancelled.

7.1.2 Example 1

Use language similar to the following when reuse of software components will be considered as part of the Government's competitive selection criteria:

> The Government will consider the offeror's ability to successfully integrate existing software components into its proposed architecture.

Use language similar to the following when an objective of the program is to create reusable software components:

> The offerer's understanding of reuse concepts and its approach toward design, development and testing of reusable software components will be evaluated

Use language similar to the following when the Government must baseline software rights and copyright issues as part of the competitive selection process. Note, the Government cannot require a contractor to relinquish rights for components developed exclusively at private expense. Thus, any such referral would have to be considered in terms of risk to the program, if any, in achieving objectives.

> The Government will consider the offerer's (and its subcontractors') positions regarding software and data rights and copyright, and the Government's resulting ability to attain the reuse objectives described in this solicitation.

Use the following criteria to evaluate the reusability of newly developed components (are they reusable to the Government?):

Potential for reuse with other Government requirements

Degree to which the component is modifiable

Application to other uses

Quality of the component

Isolation of the classified components

Degree to which components exceed minimum requirements

Potential for cost savings, based on cost/benefit analysis

Use the following criteria to evaluate the extent that existing components are reused in a system:

Degree that software components are reused that exceeds what was proposed (components developed under this effort as well as from other sources)

Improving upon proposed software schedule due to reuse, without negatively impacting the overall contract cost or schedule

Improving upon proposed software schedule due to reuse, without negatively impacting the overall contract cost or schedule

7.1.3 Example 2

The contractor's understanding of and experience in software development, including reuse, shall be considered in evaluation of contract proposals. Proposals will be evaluated in accordance with the factors listed below [15]:

Note: In streamlined source selection, you will want to consolidate these factors to the absolute minimum number necessary.

Technical Factors (Software and Reuse)

- Software technical approach
- Risk minimization approach
- Performance base lining and control
- Software reuse approach

Management Factors (Software and Reuse)

Software management approach

Program, resource, risk and subcontract management and control

Teamwork and interdisciplinary communications

Software configuration management

Software quality management

Software reuse library management

Types and suitability of metrics for program measurement and control of reuse activities

Levels of software rights proposed and their consistency with program reuse objectives

Management commitment/support for reuse

Cost

Realistic estimates for reuse tasks

Appropriate models and/or estimating techniques used to substantiate reuse task estimate

Risk associated with achieving reuse cost targets/budgets

7.2 Proposal Instructions (Section L)

7.2.1 Example 1

When reuse of software components is required, the following, or similar, language should be used to specify information required in the offer's proposal:

> The proposal should describe the offer's understanding of the software component, to include how the offer assesses its utility for the program described in this solicitation and its suitability for the offer's software architecture. Specific discussion of the following is required:
>
> Techniques used to validate the software component with respect to functionality and documentation
>
> Method(s) for integrating the software component(s) into the offer's design
>
> Unique performance and/or risk (technical, schedule, cost) issues associated with reuse and plans to mitigate those risks
>
> Program-specific issues affected by reuse, such as reliability and/or maintainability

Software standards proposed for use

Prototype plans/processes/techniques

Identification of sources for components

Experience in reusing existing software components

When the solicitation requires development of reusable software, language similar to the following should be used in the proposal instructions:

> The proposal will describe the offerer's software development methodology and how that methodology is suitable for creation of reusable software components. The offerer's Software Engineering Environment tools to be used in the creation, configuration control, and test and integration of the reusable components will also be described. Specific discussion of the following is also required:
>
> Any risk (technical, cost, schedule) occasioned by the requirement for development of reusable components
>
> Impact, if any, on the performance of the system
>
> Limited experience in designing and implementing reusable software components
>
> Methods for controlling configuration of the reusable component(s) to ensure common user baselines are controlled
>
> Immature software metrics and standards to be used
>
> Program-specific issues negatively impacted by reuse
>
> Immature component evaluation methods/criteria

When the effort requires domain requirements planning, the following, or similar, language should be used to identify required information:

> The proposal should describe the domain analysis approach and processes that will be used. Specific discussion of the following is also required:
>
> Domain analysis approach to be taken
>
> Domain modeling techniques
>
> How domain knowledge will be represented
>
> Tools/environment to be used in capturing domain knowledge
>
> Experiences in aspects of domain requirements planning
>
> Domains or application areas of expertise
>
> Any associated risks and plans for mitigation

When reuse-related documentation is required, language similar to the following should be used to identify information required in the offerer's proposal:

a. Software Reuse Plan

The Software Reuse Plan shall identify and document the objectives, methods and criteria for creating, acquiring, maintaining and evolving reusable software components. This plan shall be an integral part of the Software Development Plan. The Software Reuse Plan shall describe how software products and components will be identified and developed as candidates for reuse. It shall describe procedures to be used to ensure that reusable software components are developed in accordance with the [specify standards]. It shall comply with the Software Reuse Strategy of the system.

b. Software Library Report

The contractor shall develop a Software Library Report to document the choice of a domain-specific (horizontal or vertical domain) reuse library. The report should identify:

Program's library requirements

Criteria used in evaluating repositories

Libraries that are being evaluated

Evaluation results

Choice of library

c. Library Development and Management Plan

The Library Development and Management Plan shall document the proposed plans, procedures and methods to establish a domain-specific library. Specific discussion of the following is also required:

Library mechanism(s) and classification scheme(s)

Domain engineering approach

Library support tools

Library environment (hardware, software)

Library environment (hardware, software)

Experience in using/developing/managing various libraries

User support

Any associated risks and plans for mitigation

Domain knowledge management

Component management

The solicitation must describe the Government's perceived minimum needs for software and data rights and copyright. The instruction for the proposal should require offerers to describe how they propose to satisfy those needs. The Government Program Office should also consider requesting alternative approaches to ownership and rights which would still satisfy the Government's needs, while potentially offering more protection/incentive to industry. Language similar to the following could be used:

> The Government believes it requires (unlimited/restricted) rights in software and (unlimited/limited) rights in data to successfully implement the reuse objectives of this solicitation. The Government (also requires/does not require) assignment of copyright for these same objectives. The offerer should carefully review the requirements of this solicitation, including DFARS 252.227-7013 and the provisions regarding predetermination of rights, and identify in the proposal its and its subcontractors'/vendors' intent to comply with these reuse requirements. Should any software or supporting data be proposed with less than the rights (and assignments) identified above, the offerer will specifically identify that software and/or data, indicating the rights with which it will be provided and the basis for claiming those rights. Additionally, the offerer will identify how the Government may still satisfy its reuse requirements within the proposed rights position(s).

7.2.2 Management Proposal Information for Reuse Approaches

The offerer will identify its software management approach and how software reuse activities are integrated. Reuse management techniques will be characterized as either standard to the offerer's set of practices or unique to this effort. All reuse practices will be substantiated as to their appropriateness and utility to this effort. The offerer will address, at a minimum:

- Configuration management practices
- Risk procedures, controls and abatement techniques for reuse
- Application of concurrent engineering, integrated product development, or similar techniques to reuse activities
- Types of metrics used, history of the offerer's use of these metrics and their utility for this reuse work
- Methods for establishment and/or use of libraries/repositories

The offerer will also specifically identify any and all software components proposed or to be developed which will be delivered with less than unlimited rights. Specifically, the offerer will identify its subcontractors'/vendors' positions with respect to restricted software rights, any associated limited rights data, copyright and any patented software. A format similar to the following, Table 7-1, Impacts of Data Rights, will be used.

Table 7-1 Impacts of Data Rights

	Program Impact	Offeror Commercial Impact
1a. Restricted Rights Software		
1b. Limited Rights Software Data		
2. Copyright 1. a. Contractor Retention 2. b. Government Assignment		
3. Patent 1. a. Existing 2. b. Pending		

The offerer will include supporting rationale for any proposal to provide the Government any rights on a deferred delivery or deferred ordering basis, or any other basis which is different from theGovernment's Proposed position in this solicitation.

7.2.3 Cost Proposal Information

The offerer will identify all models and/or other cost estimating techniques used to estimate and size the total software effort. Specific reuse discussion will be provided, addressing:

- Tuning of models for reuse
- Development and/or use of unique models for reuse

  Reuse of existing components "as is"

  Development of reusable components

  Modification of existing components to make them reusable

The basis of estimates furnished will specifically address and segregate reuse estimates to enable the Government to determine whether these costs are realistic and reasonable.

7.2.4 Software Reuse in Requests for Proposal

As part of the Technical Proposal, the Contractor shall submit a section entitled "Software Reuse Plan." This section shall describe the Contractor's plan for implementing software reuse as an integral part of the development of the prospective system.

The plan should indicate the expected objects and levels of reuse. This reuse may include, for instance, the reuse of requirements, design of algorithms or data structures, and documentation as well as code. Further, software reuse can occur at many levels, e.g., CSCI, CSC, CSU.

The contractor's plan should explain how the contractor is addressing the problem of pinpointing potentially reusable functions, such as CSCIs, CSCs and CSUs. The contractor should identify any candidate software functions that it considers to be potentially reusable.

The plan should describe the sources of reusable requirements, design documents and software components, etc., that the contractor has investigated for potential reuse, and others whose investigation is planned. For software components, these sources should include public domain and Government libraries, commercial vendors, and in-house libraries.

The plan should describe how the contractor will allot its time in the prospective schedule to determine the candidate lists of reusable objects and sources of reusable software.

The plan should describe the day-to-day policies that the contractor will follow to implement software reuse. The contractor's policy might include:

A methodology for the evaluation of reusable software items This methodology might specify a set of criteria and how to apply them. It might also prescribe the use of certain tools to implement this methodology.

A set of programming guidelines established by a company to promote reuse in its various forms. There might also be a set of guidelines on how best to integrate reusable code with newly written software for a proposed system. An account of these guidelines and their development, as well as the Contractor's experience with them, might be useful.

Delineation of personnel (management and staff) whose responsibility is the implementation of reuse activities throughout the company in general. The plan might explain how these personnel are expected to promote reuse on this project in particular.

The plan should describe the expected benefits to be derived from its implementation. These benefits may include cost savings, manpower reductions, risk reduction, increased software quality, and shorter development time. It might also describe the benefits that the contractor has gained from past implementation of its policies and the benefits expected in the future. In addition, it might list the benefits expected for the Contracting Agency during system development and at later stages of the prospective system's life cycle.

Finally, the contractor shall establish milestones to implement this Software Reuse Plan. These milestones shall provide a basis for determining the extent to which the project is following the stated plan, and an opportunity to revise it as necessary.

Upon contract award, this plan shall be incorporated into the Software Development Plan (SDP), and revised and updated in accordance with the established software milestones.

7.2.5 Proposal Instructions for Reusable Software Components

Any software or software documentation which is identified as restricted rights software shall be specifically listed in the contract, in accordance with DFARS 252.277-7013.

If license agreements are not negotiated and made part of this contract, unlimited rights shall be applicable to all components not identified as having been developed at private expense.

Components delivered under this contract shall be entered into a reuse library. There shall be no additional warranty responsibility after entry into the library (other than any warranty which was required in the contract that covered original development). Components should be certified and should include the results of the certification process and indicate the tools that were used to perform the certification process.

DFARS 252.227-7026 dealing with deferred delivery of technical data or computer software, and DFARS 252.227-7027 dealing with deferred ordering of technical data or computer software, apply to this contract, and may be specifically applied to components.

The contractor shall provide a list of all software objects to be delivered with restricted rights, in accordance with DFARS 252.277-7013 and DFARS 252.227-7019.

The contractor shall submit a draft Software Reuse Plan (SRP) which describes current efforts that facilitate software reuse, and describes examples of reuse projects and a proposed approach for application on this contract.

7.2.6 Rights In Computer Software and Computer Software Documentation

Taken from the All Source Analysis System RFP: Army Tactical Command and Control System

It is contemplated that the vast majority of the applications computer software delivered under this contract will be developed hereunder and, accordingly, will be furnished, along with its documentation, with unlimited rights. For that deliverable software which is not so categorized and is intended to run on the CHS target machines, it is the intention of the Government to enter into a licensing scheme that meets the Government's fielding concepts and requirements, accommodates commercial software licensing practices, and is consistent with regulatory requirements. Toward that end, all software licenses proposed for this software should meet the minimum requirements of DFARS 252.227- 7013 and the additional requirements set out below.

a. The right to use the software on any ACCS computer or in any ACCS and/or ASAS software development or support center whether Government or third party operated.

b. The right to make and use any derivative works of the computer software and computer software documentation for use on, or with, any ACCS computer or in any ACCS and/or ASAS software development or support center whether Government or third party operated.

c. The right to license the software for Government-owned computers other than those supplied under this contract, at the best commercial rate available at the time of the licensing need.

d. The right to provide access to, and use of, the software and documentation, by third parties under contract with the Government, in support of, or for use on, the ACCS and/or ASAS programs, provided appropriate confidentiality agreements are executed.

e. The right to make an unlimited number of copies of the software and all related documentation for use on all licensed ACCS computers or any subset thereof. This copying should be permitted without the necessity of a separate accounting.

With respect to that software furnished which is intended to run only on PDSS machines, all software licenses proposed should meet the minimum requirements set out in DFARS 252.227-7013 and the additional requirements set out below:

a. The right to use the computer software on or with any licensed ASAS and/or ACCS related software development or support center, whether Government or third party operated. The contractor agrees to license as many of such centers as the Government feels necessary, at the best commercial rate available at the time of the licensing need.

b. The right to make and use any derivative works of the computer software and computer software documentation on, or with, any licensed ASAS or ACCS computer, provided appropriate confidentiality agreements are executed.

c. The right to provide access to, and use of, the software and documentation, by third parties under contract with the Government, in support of, or for use on, the ACCS and ASAS program, provided appropriate confidentiality agreements are executed.

Documentation relative to this software will be furnished with "limited rights" as defined in DFARS 252.227-7013, with the additional right to disclose the documentation to third party support contractors provided appropriate confidentiality agreements are executed.

With respect to that software furnished which is intended to run only on the interim (non CHS) target machines, all software licenses proposed should meet the minimum requirements set out in DFARS 252.227-7013 and the additional requirements set out below:

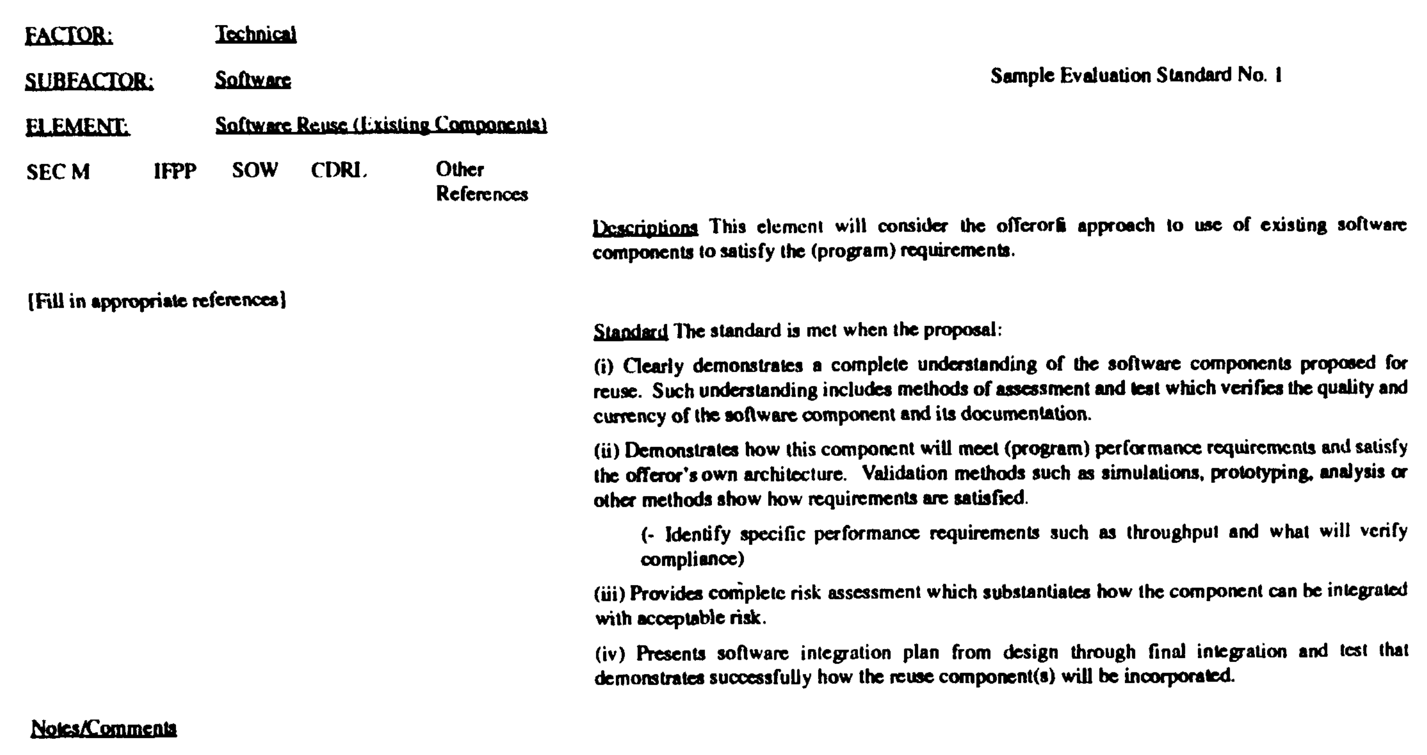
a. The right to use the software and make and use any derivative works thereof, on any interim target machine furnished under this contract and on any ACCS or ASAS software development or support center, whether Government or third party operated, provided appropriate confidentiality agreements are executed.

Documentation relative to this software will be furnished with "limited rights" as defined in DFARS 252.227-7013, with the additional right to disclose the documentation to third party support contractors provided appropriate confidentiality agreements are executed.

Documentation relative to this software will be furnished with "limited rights" as defined in DFARS 252.227-7013, with the additional right to disclose the documentation to third party support contractors provided appropriate confidentiality agreements are executed.

7.3 Sample Evaluation Standards

Evaluation standards are needed to objectively assess offerer responses to reuse requirements. Figures 7-1, -2 and -3 are sample evaluation standards for each of the paragraphs described under Section M, Evaluation Criteria.



Sample Evaluation Standard No. 1

<u>FACTOR:</u> <u>Technical</u>

<u>SUBFACTOR:</u> <u>Software</u>

<u>ELEMENT:</u> <u>Software Reuse (Existing Components)</u>

SEC M	IFPP	SOW	CDRL	Other References

[Fill in appropriate references]

<u>Descriptions</u> This element will consider the offerorß approach to use of existing software components to satisfy the (program) requirements.

<u>Standard</u> The standard is met when the proposal:

(i) Clearly demonstrates a complete understanding of the software components proposed for reuse. Such understanding includes methods of assessment and test which verifies the quality and currency of the software component and its documentation.

(ii) Demonstrates how this component will meet (program) performance requirements and satisfy the offeror's own architecture. Validation methods such as simulations, prototyping, analysis or other methods show how requirements are satisfied.

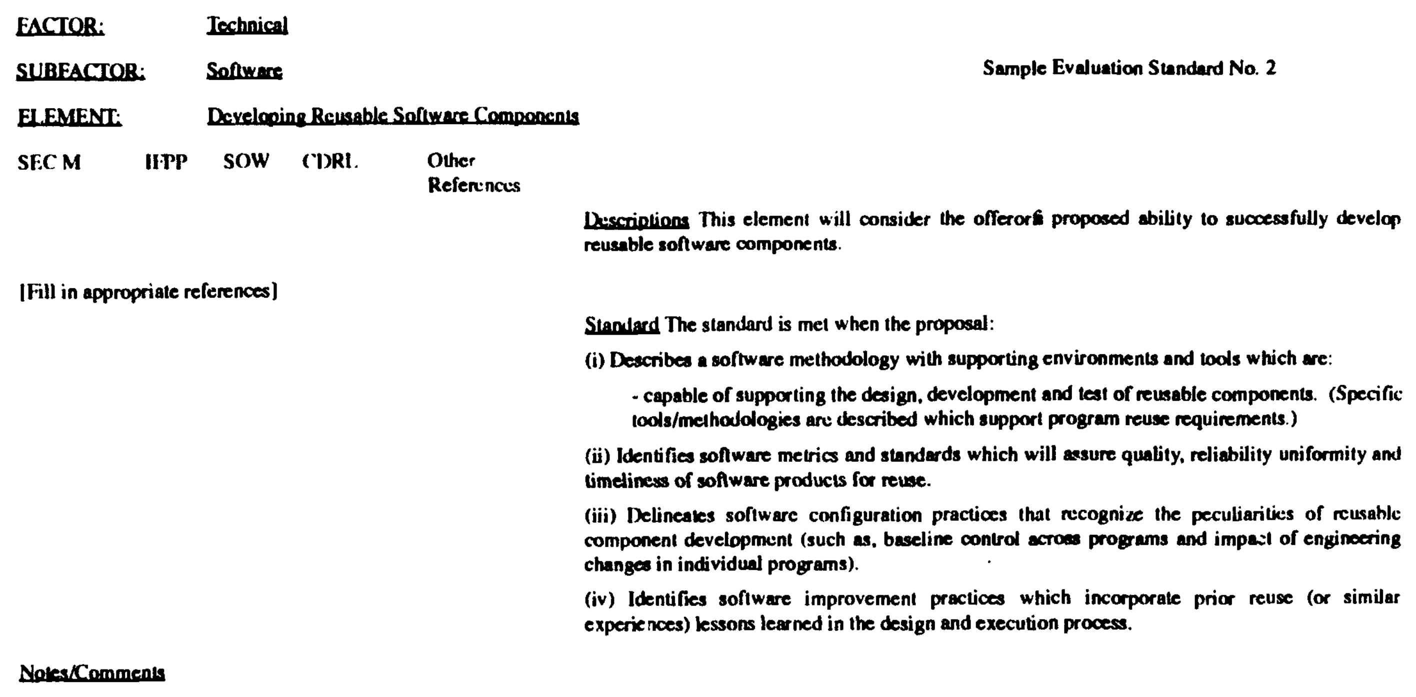
(- Identify specific performance requirements such as throughput and what will verify compliance)

(iii) Provides complete risk assessment which substantiates how the component can be integrated with acceptable risk.

(iv) Presents software integration plan from design through final integration and test that demonstrates successfully how the reuse component(s) will be incorporated.

<u>Notes/Comments</u>

Figure 7-1 Sample Evaluation Standard No. 1



Sample Evaluation Standard No. 2

<u>FACTOR:</u> <u>Technical</u>

<u>SUBFACTOR:</u> <u>Software</u>

<u>ELEMENT:</u> <u>Developing Reusable Software Components</u>

SEC M	IFPP	SOW	CDRL	Other References

[Fill in appropriate references]

<u>Notes/Comments</u>

<u>Descriptions</u> This element will consider the offerorß proposed ability to successfully develop reusable software components.

<u>Standard</u> The standard is met when the proposal:

(i) Describes a software methodology with supporting environments and tools which are:

- capable of supporting the design, development and test of reusable components. (Specific tools/methodologies are described which support program reuse requirements.)

(ii) Identifies software metrics and standards which will assure quality, reliability uniformity and timeliness of software products for reuse.

(iii) Delineates software configuration practices that recognize the peculiarities of reusable component development (such as, baseline control across programs and impact of engineering changes in individual programs).

(iv) Identifies software improvement practices which incorporate prior reuse (or similar experiences) lessons learned in the design and execution process.

Figure 7-2 Sample Evaluation Standard No. 1

Sample Evaluation Standard No. 3

FACTOR:	Management
SUBFACTOR:	Reuse Rights
ELEMENT:	N/A

SEC M	IFPP	SOW	CDRL	Other References

[Fill in appropriate references]

Notes/Comments

Descriptions This subfactor will assess the offerorß compliance with Government requirements for software and data rights (and copyright).

Standard The standard is met:

(i) When the proposal offers at least the minimum rights (and copyright assignment) required, as stated in RFP Section ____.

or

(ii) Where other rights are described, the offerorß plans for satisfying reuse requirements within these alternative rights are adequate.

Figure 7-3 Sample Evaluation Standard No. 3

7.4 Cost Framework

The decision to reuse existing components is made in consideration of many factors, but cost/ benefit is clearly one of the most important of them. For a manager to want to reuse existing components, there should be a perception of cost savings (unless schedule savings or some other program parameter has been chosen as the key benefit). In making the reuse decision, the following variables must be considered:

Cost (Reuse) =	There is a cost incurred when reusing an existing component; this includes the cost to identify, retrieve, understand, validate, integrate and test a component.
Cost (Modify) =	In some cases, it is necessary to modify the validated component before use in the current application; this is the cost incurred for such modification.
Cost (Incent) =	There is an incentive paid to the original developer of a component when it is reused; there can also be an incentive paid to the current developer to motivate him to use the existing component rather than develop a new one on his own; this variable includes the cost of any royalties or license fees; these incentives are part of the cost calculation (verify legitimacy payments).
Cost (Devel) =	If reuse were not possible, this would be the cost associated with designing and developing the component anew for one-time use.

The net savings (NS) to the user can be calculated using the following formula:

NS = Cost (Develop) - Cost (Reuse) - Cost (Modify) - Cost (Incentive)

For example, if the cost to develop a new component is $100,000 and the cost of reuse (i.e., identification, retrieval, understanding, validation, integration and test) is $50,000, the incentive paid to the original developer is $5,000 and the incentive paid to the current developer is $5,000, and there is no modification required to the existing component, then the net savings is $40,000 ($100,000 - $50,000 - $5,000 - $5,000), and reuse in this case makes good economic sense. If, on the other hand, there is modification required and that modification will cost $50,000, then the net savings will be a negative $10,000 ($100,000 - $50,000 - $50,000 - $5,000 - $5,000), indicating that reuse does not, in this case, make economic sense (however, it may still be an objective for other than economic reasons).

Program managers are also tasked with developing components specifically for reuse. It is recognized that this development will cost the program extra time and money because of the considerations discussed in Section 2.4.3. The manager may want to know how many instances of reuse it would take to recoup the extra development dollars spent. In this case, the following variables apply:

Cost (Devel+) =	This is the cost associated with designing and developing the component anew, given that it will be reused by other applications later; this cost is more than Cost (Devel) above.
Cost (Deposit) =	This is the cost associated with adding a component to the library.
Cost (Maint+) =	This is the cost associated with maintaining the reusable component in the library and includes configuration management and change distribution.
NS (Net Svgs) =	This is the net savings to each subsequent user of the reused component as calculated by the formula discussed above.
QTY =	This is the number of subsequent users of the reusable component.

To calculate the number of subsequent users (i.e., QTY) that are needed to recoup the additional investment when developing for reuse, the following formula applies:

NS * QTY > [Cost (Devel+) - Cost (Devel)] + Cost (Deposit) + Cost (Maint+)]

QTY must be high enough to make the left-hand side of the above expression greater than the right-hand side. For example, if Cost (Devel+) is $125,000 and Cost (Devel) is $100,000, Cost (Deposit) is $10,000 and Cost (Maint) is $40,000, and NS is $25,000, then there must be 3 subsequent users to recoup the additional $75,000 in cost ($125,000 - $100,000 + $10,000 + $40,000).

7.5 Award Fees

7.5.1 Award Fee Example

The following is an excerpt from the Advanced Field Artillery Tactical Data System (AFATDS) Award Fee Determination Plan, 26 Dec 91, Revision L.

AWARD FEE DETERMINATION PLAN
SECTION I
INTRODUCTION

1. Purpose
The purpose of this plan is to prescribe the responsibilities, procedures, definitions and guidelines for assessing _____ performance on the AFATDS Version 1 software development in order to determine award fee. This plan will serve as a baseline for the Government and to understand the basis for award fee determinations.

2. Capability
This plan, together with related contractual provisions, will be used by the Government personnel involved in the award fee evaluations during the Version 1 software development contract and any award fee options to that contract that are exercised. This plan may be revised, as necessary, to reflect exercised options.

3. Definitions
a. Award Fee. A variable fee which is determined by Government evaluation of the contractors' performance during specific award fee periods. The total amount of award fee will be determined at contract negotiations and included in the contract. The amount of the award fee available at each milestone (as defined by DoD-STD- 2167A) or Award Fee Period will be as follows:

Five months after contract award	5%
Completion of SSR	15%
Six months after last SSR	20%*
Completion of build 1	20%
Completion of Final CDR	10%
Government acceptance of system SW	25%
IOT&E + 30 days	5%
Total	100%

* Completion of Build 1 is defined as that point at which the Build 1 software is released by the contractor's Software Engineering to the contractor's Independent Test Organization (ITO). The contractor will notify the Government approximately two weeks in advance of the event and, if necessary, the event will be confirmed by the Government In-Plant Working Group in conjunction with the local DSACPRO.
b. Award Fee Determining Official (AFDO). The designated (by Principle Assistant Responsible for Contracting) Government official who will determine the amount of award fee to be paid at each award fee period. The AFDO will be the PM FATDS.
c. Award Fee Review Board (AFRB). Government personnel, appointed by the AFDO, who are responsible for determining evaluation factors prior to each award fee period and assessing performance in terms of those factors. The AFRB recommends to the AFDO the award fee based upon evaluation of the contractor's performance.
d. Award Fee Evaluator. Individual, appointed by the AFDO, who serves on the AFRB and is responsible for evaluating contractor performance in a specific area and to present his evaluation to the AFRB.
e. Contractor. The term "contractor" and ____ are used interchangeable in this plan.

4. Approach
a. The award fee earned shall be determined subjectively by the government based upon accomplishment of elements listed and defined in Appendix A to this plan.

b. The initial award fee evaluation criteria percentages for the initial three periods will be established at the time of contract award. The award fee evaluation criteria percentages for the remaining periods will be reflected in the contract as planning figures. (See Appendix C for proposed award fee criteria percentages.) Sixty (60) days prior to completion of the Six Months After Last SSR Evaluation Period, the contractor and the Government will negotiate award fee evaluation criteria percentages for the remaining periods and these percentages will be reflected in a contract modification. Should the parties fail to achieve agreement on the remaining percentages, the Government will issue a unilateral modification to the contract to prescribe award fee criteria percentages. This modification will be subject to the Disputes Clause.

SECTION II
ORGANIZATION, RESPONSIBILITIES AND PROCEDURES

1. Organization Structure
The AFRB will consist of Award Fee Evaluators for each element listed in the Appendix A and will be chaired by the Product Manager, AFATDS. If an element is not to be evaluated at a particular award fee period, the evaluator will not be part of the AFRB for that review.

2. Responsibilities
a. The overall responsibility of the Award Fee Evaluators is to adequately, honestly and accurately evaluate and record the evaluation to insure that the AFRB and AFDO are provided sufficient information to determine fee. The Evaluator will evaluate the quality of work as demonstrated by the contractor of his technical competence, ability to apply that competence to the project and the contractor's overall quality of products delivered (including ease of understanding, clarity and use of correct English).
b. The AFRB, as a body, will review individual evaluations and calculate a recommended award fee for presentation to the AFDO.
c. The Chairman of the AFRB, the PM AFATDS, will:

1. Coordinate with Award Fee Evaluators and the AFDO to determine the elements to be evaluated during rating periods beyond PDR and participate in negotiation of proposed award fee evaluation percentages.

2. Call the AFRB together after completion of award fee periods to determine fee recommendations.

3. Brief the AFDO on AFRB findings and recommendations.

3. Procedures
a. Sixty days prior to the completion of the PDR Award Fee period, the Procuring Contracting Officer (PCO) and the Chairman of the AFRB, will, in coordination with the Award Fee Evaluators and the AFDO, negotiate the elements to be evaluated during the remaining award fee periods and their relative weights for determining award fee amounts. One of the inputs to determining elements and relative weights will be performance during the previous rating periods.

b. Within five working days after the completion of the award fee period, the Award Fee Evaluators will prepare written reports evaluating the element for which they are responsible. Evaluations will be based upon documentation produced by the contractor (e.g., CDRLs), performance of contractor during major milestone reviews, input from sources outside of the Evaluator (e.g., other Army/USMC personnel) and personal observations and evaluation of the contractor's performance by the evaluator. The contractor will be invited, by the Chairman of the AFRB, to provide written input to the AFRB as to how he believes he has performed in each element in the period. The Chairman will request this input be provided at the time of completion of the award fee period. It will be provided to each evaluator for consideration.
c. Within ten working days after the completion of an award fee period, the Chairman will conduct a formal AFRB to determine total recommended fee.
d. Within fifteen days after the completion of an award fee period, the Chairman of the AFRB will present the recommended award fee including narrative justification and contractor's input to the AFDO.
e. Within twenty working days after the completion of the award fee period, the AFDO will furnish written notification, through the PCO, to the contractor advising of the recommended award fee. The recommended award fee will be based upon the AFRB recommendation and the AFDO's independent review and analysis of program status and contractor's performance. The notice will include a summary supporting the award fee together will specific reasons for any low ratings which may have been recommended. Within fifteen calendar days after receipt of the award fee determination, the contractor may submit written comments/rebuttal to the AFDO through the PCO.
f. Within fifteen working days after the PCO notification to the contractor, or ten days after receipt of contractor comments, whichever is later, the AFDO will determine the amount of any award fee to be paid to the contractor for the period under consideration.
g. Within fifteen work days after the receipt of the award fee determination from the AFDO, the PCO will issue a unilateral contract modification setting forth the specific amount of award fee determined to have been earned. The contractor will also be provided an explanation of the Government's assessment of his performance for the applicable period if any changes were made from the initial summary previously provided. The decision as to the amount of the award fee shall not be subject to the Disputes Clause of the contract.
h. The contractor may submit a voucher for that portion of the award fee to which entitled no sooner than fifteen calendar days after the above modification.
i. At any time during the performance of the contract, the contractor will be notified in writing of areas of deficiency in order to afford him the opportunity to improve his performance.
j. Award Fee available but not earned during an evaluation period will, at Government discretion, be made available during the last two rating periods. However, at a minimum, 10% of the unearned fee will be added to the award fee pool for the Government Acceptance of System Software Period. The remaining available, unearned fee, may be applied in any proportion to the two last award periods. The application of unearned fee (less the 10% described above) and the periods and criteria to which it will be applied will be at Government discretion.
k. The purpose of making unearned fee available at the end of the contract would be to encourage the contractor to correct early problems through aggressive management. Therefore,

the Government will notify the contractor as early as possible of its intention concerning available, earned fee. At the latest, the contractor will be notified at least ninety (90) days prior to the completion of the award fee period for the Government Acceptance of System Software of the amount of unearned fee that will be made available for the remaining periods and criteria to which it will be applied.

l. When the Cost Plus Award Fee portion of the contract is modified which results in adding or deleting funds in the Award Fee Pool, that change in funding will be pro-rated over the current and future Award Fee periods. No retroactive changes will be made to funding for Award Fee periods that have already passed.

SECTION III
EVALUATION CRITERIA

1. Evaluation Elements

The elements for evaluation are divided into four categories: Overall Technical and Programmatic Management, Technical, Cost and Schedule. Provided below are the elements:

Overall Technical and Programmatic Management

Technical

- Optimum Utilization of Common Hardware and Software (CHS)
- Overall Logistics Management
- Implementation of Ada
- Embedded Training
- System Performance
- Software Quality Process
- Software Quality
- Software Reusability
- Tests
- MANPRINT

Cost

- Overall Program Cost

Schedule

- Contract Deliverables
- Program Schedule

A complete definition of these elements is provided in Appendix A.

2. Rating Criteria
Each evaluator will assign a rating to each element, as appropriate, and compute an overall element rating. Appendix A provides a definition of "Marginal", "Good" and "Exceptional". The numerical rating for these adjectives are:

Numerical Rating	Adjective Rating
71 - 100%	Exceptional
41 - 70%	Good
1 - 40%	Marginal
0%	Unacceptable

3. Recommended Award Fee Evaluation Formula

a. The award fee will be calculated as follows:

1. Evaluator will select adjective rating for each element being evaluated
2. Select numerical rating within range of selected adjective rating
3. Multiply total element rating by dollars available in that element to determine fee for each element.
4. Add all element fees together to determine total award fee

b. Example: Total fee available for period is $1,000,000. Weight of MANPRINT for period is 20%

1. Available Fee for Element - 20% x $1,000,000 = $200,000
2. Element Rating - Exceptional
3. Element Numerical Rating - 86%
4. Element Fee - 86% x $200,000 = $172,000
5. Total Fee:

   Other Elements = $710,000

   MANPRINT = $172,000

   Total = $882,000

6. In this example, $882,000 would be the recommended fee for the period.

APPENDIX A

EVALUATION ELEMENTS

The evaluation elements and their adjectival rating are delineated below.

Overall Technical and Programmatic Management

1. Overall Technical and Programmatic Management. This includes the contractor's management system which encompasses program deliveries; technical personnel management; technical reporting; technical/programmatic reviews; pro-active technical management; visibility into technical progress; risk management; accuracy, timeliness and utilization of Software Management Quality Indicators ("Metrics"); and communications and involvement with the Government that supports and sustains an atmosphere of cooperation and team effort.

   a. **Marginal** - Contractor's management allows programmatic/technical issues under his control to disrupt the program to some extent; however, the contractor demonstrates a capability to manage the majority of the problems without Government intervention.

   b. **Good** - Contractor's management is capable of resolving programmatic/technical issues which are under his control. They are resolved through pro-active advance planning and good cooperative communications with the Government and thereby have no adverse impact on the program.

   c. **Exceptional** - Contractor's management is capable of anticipating programmatic/technical problems within his control and planning for them in cooperation with the Government. The contractor takes positive action to eliminate incipient problems before they have an impact on the program.

Technical

1. Optimal Utilization of Common Hardware and Software (CHS). This includes maximized use of ATCCS common hardware and software without adverse performance on the AFATDS system.

   a. **Marginal** - The contractor demonstrates understanding of the CHS, but no innovation is demonstrated in the integration of the CHS into AFATDS.

   b. **Good** - Contractor demonstrates a good understanding of the CHS and demonstrates initiative in the integration of the CHS to achieve system performance requirements.

   c. **Exceptional** - Contractor demonstrates an excellent understanding of the CHS and demonstrates outstanding initiative in the integration of the CHS to achieve system performance.

2. Overall Logistics Management. This includes an effective training program to support tests and fielding, and acceptable logistical documentation.

   a. **Marginal** - The contractor training personnel meet the standards of the Statement of Work, Annex E, Para. 6.3 (Training of Instructors and Key Personnel). Training is acceptable with some flaws, but is supportive of Government comments, accurately represent the product and are written to the proper grade level, such that they can be/have been validated and verified.

   b. **Good** - The contractor training personnel meet or exceed the standards of the Statement of Work, Annex E, Para. 6.3 (Training of Instructors and Key Personnel). Training is well organized and is supportive of all Government testing and subsequent fielding. Portions of the technical publications produced by the contractor are very responsive to Government concepts, are complete, accurate and show consistency in terminology and can be understood and used by personnel for which they are intended.

   c. **Exceptional** - Contractor training personnel surpass the standards of Annex E, Para. 6.3 (Training of Instructors and Key Personnel). Training is exceptionally well prepared; innovative and presented to maximize support for testing and subsequent fielding. Portions of the technical publications prepared by the contractor are of a very high caliber such that they are complete, accurate and well organized with the appropriate level of detail and art work to make them fully understandable and easy to use at the grade level for which they are intended.

3. Implementation of Ada. This includes implementation of Ada in accordance with the Government accepted Standards for Ada Language Coding, SGGM-R551, Vol 1, December 22, 1989, AFATDS Attachment 1, dated ____.

   a. **Marginal** - The contractor generally adheres to the Standard cited above.

   b. **Good** - The contractor rigorously adheres to the majority of the Standard cited above.

   c. **Exceptional** - The contractor rigorously adheres to all portions of the Standard cited above.

4. Embedded Training. This includes ease of operation of the embedded training by the operator, effectiveness of embedded training in maintaining operator proficiency and completeness of operator documentation of embedded training.

a. **Marginal** - Contractor's implementation of embedded training does not require additional hardware; however, embedded training is awkward for the operator to use. The quality of the embedded training is adequate for the operator.

b. **Good** - Contractor's implementation of embedded training does not require additional hardware and is user friendly. The quality of the embedded training for use by the operator is good.

c. **Exceptional** - Contractor's implementation of embedded training does not require additional hardware and is user friendly. The quality of the embedded training for use by the operator is superior.

5. System Performance. This includes contractor's delivered system which is measured against the criteria (less those for which waiver was granted) enumerated in Appendix 2 to the A Spec as further defined in Government approved Test Procedures. It also includes the user representative (Army and USMC) evaluation of the usability of the system to the target force.

   a. **Marginal** - Delivered system meets performance thresholds. User finds system acceptable, but has some difficulty with the system.

   b. **Good** - Delivered system exceeds some mission critical thresholds. User finds system easy to use, has improved FSC^2 capability, but requires some minor adjustments.

   c. **Exceptional** - Delivered system significantly exceeds some mission critical thresholds. User finds system easy to use and there are no significant improvements to the FSC^2 capability.

6. Software Quality Process. This includes effectiveness of contractor's internal quality management program to include the contractor's audit processes.

   a. **Marginal** - Contractor's software quality program has minimal positive impact on the quality of the software documentation delivered.

   b. **Good** - Contractor's software quality program has a positive impact on the quality of the software documentation delivered.

   c. **Exceptional** - Contractor's software quality program has significant positive impact on the quality of the software documentation delivered.

7. Software Quality. This includes only quantifiable properties of the AFATDS software as measured by the Ada MAT Version 2.0. The metric elements measured will be accumulated under the following six criteria: ANOM[illegible]Y Management, Inde-

pendence, Modularity, Self Descriptiveness, Simplicity and System Clarity. Typical scores shall be as identified in the "Report on AFATDS Source Code Analysis Using Ada MAT Measurement Tool (draft)", TAM No. B0156-126, TFSFMD 89-1345.

a. **Marginal** - Averages over all new developed AFATDS software (except that code excluded by mutual agreement) fall within the typical range and no more than 2 of the cited criteria in the lower quarter of the typical ranges as cited in the referenced document.

b. **Good** - Averages over all new developed AFATDS software (except that code excluded by mutual agreement) for the six criteria fall in the upper halves of the typical ranges as cited in the reference document.

c. **Exceptional** - Averages over all new developed AFATDS software (except that code excluded by mutual agreement) fall in the upper halves of the typical ranges cited in the reference documents and averages for at least three (3) criteria fall in upper quartiles.

8. Software Reusability. This includes the extent to which the contractor assesses CEP code for reuse viability, the extent to which CEP code is reused in Version 1, the extent to which the contractor considers software reuse from other Ada programs, and the extent of actual software reused from other Ada programs. This also includes the contractor's ability to design Ada code that is reusable by other ATCCS Battlefield Functional Areas (BFA).

a. **Marginal** - Contractor as assessed 100% of CEP code for reuse in Version 1. Contractor has reuse 60% of CEP code assessed as being viable. Contractor has investigated potential reuse of software which will be available from some of the other Ada programs. Contractor reuses some of the code assessed as practical for reuse and available from other Ada programs. Contractor designs software which may have some reuse in other BFAs.

b. **Good** - Contractor has assessed 100% of the CEP code for reuse in Version 1. Contractor has reused 80% of CEP code assessed as being viable. Contractor has investigated potential reuse of software which will be available from other Ada programs. Contractor reuses significant amount of code assessed as being practical for reuse and available from other Ada programs. Contractor plans for and identifies software components which will be designed for reuse by other BFAs; identified software components have nominal reuse capability.

c. **Exceptional** - Contractor has assessed 100% of the CEP code for reuse in Version 1. Contractor has reused 95% of CEP code assessed as being viable. Contractor has investigated potential reuse of software which will be available from other Ada programs. Contractor reuses significant amount of code assessed as being practical for reuse and available from other Ada programs. Contractor plans for and identifies software components which will be designed for reuse by other BFAs; identified software components have extensive reuse capability.

9. Tests. This includes CSCI Formal Qualification Testing and System Software Testing.

   a. **Marginal** - Contractor software passes testing with an acceptable number of priority 3 and 4 errors ad defined in the Statement of Work, Para 4.8.1

   b. **Good** - Contractor software passes testing with fewer than one priority 3 errors per 5400 Source Lines of Code (SLOC) and one priority 4 or 5 error per 2700 SLOC.

   c. **Exceptional** - Contractor software passes testing with fewer than one priority 3 error per 7200 SLOC and one priority 4 or 5 error per 3600 SLOC.

10. MANPRINT. This includes an effective MANPRINT program (i.e., human factors engineering, system safety, health hazards, manpower and personnel, excluding training).

    a. **Marginal** - Contractor has implemented a MANPRINT program that provides consideration of MANPRINT elements in the software design, integration and screen development such that it provides operator proficiency that is easily obtained and maintained.

    b. **Good** - The contractor has planned a MANPRINT program that is implemented early in the system engineering phase to significantly influence software design, integration and screen development such that it provides a system that is easy to train and operate significantly reducing the cognitive load on the operator.

    c. **Exceptional** - The contractor has planned and implemented a MANPRINT program that is a driving influence during software design, integration and screen development utilizing innovative techniques that provide a system which is exceptionally designed with the soldier in mind, helping him to learn and use the system.

Cost

1. Overall Program Cost. This includes the contractor's management of costs for activities performed in accordance with the negotiated contract. It includes the contractor's ability to aggressively review cost trends and take appropriate action to control costs. A variance caused by the differences between negotiated indirect rates

and actual indirect rates will not be considered a contractor caused unfavorable variance.

a. **Marginal** - As reflected in the delivered Cost Performance Reports (CPR), contractor's Actual Cost of Work Performed (ACWP) and Budgeted Cost of Work Performed (BCWP) during the period have few significant unacceptable variances. Due to Government intervention, contractor has instituted management practices or has instituted trade-offs that should bring costs under control.

b. **Good** - As reflected in the delivered CPRs, contractor's ACWP and BCWP during the period have no significant unacceptable variances. Contractor has taken initiative to institute aggressive management practices that should bring costs under control.

c. **Exceptional** - As reflected in the delivered CPRs, the variance between contractor ACWP and BCWP during the period is acceptable. Contractor has established procedures which aggressively manage costs and cost control has been implemented as an integral part of the contractor's management approach.

Schedule

1. Contract Deliverables. This includes the contractor's delivery of data items in accordance with the schedule laid out in Contract Data Requirements List (CDRL) of the contract (delivery date is date data item is received at PM FATDS, Fort Monmouth, NJ); however, it does not apply to items requested from the Data Accession List; and that the data delivered has been determined by the Government to be suitable for the purpose intended. The term "critical data item" is used to denote data items the Government deems to be critical to the execution of the program and delay in delivery of them would impact the program. Failure to deliver a scheduled data item may, at the judgement of the Government, render this criteria rating as "Unacceptable". Failure to meet the minimal delivery requirements outlined below may also result in an "unacceptable" rating.

a. **Marginal** - In the absence of excusable delays, all critical data items scheduled to be delivered during the period are late by more than 10 working days and the majority of the other data items are delivered within 15 working days of the schedule.

b. **Good** - In the absence of excusable delays, all critical data items scheduled to be delivered during the period are late by more than 5 working days and most of the other data items are delivered within 10 working days of the schedule.

c. **Exceptional** - In the absence of excusable delays, all critical data items scheduled to be delivered during the period are on time and all other data items are delivered within 5 working days of the schedule.

2. Program Schedule. This includes the contractor's ability to maintain the total program on schedule. This includes meeting major milestones, and coordinating all contractual elements.

   a. **Marginal** - In the absence of excusable delays, major milestones within the period are delayed by at most 40 working days. Contractor schedule control does not adequately control some non-major events. Contractor, at Government behest, has instituted controls to monitor schedule, but has no hope of recovering lost schedule.

   b. **Good** - In the absence of excusable delays, major milestones within the period occur within 20 working days of planned dates. Contractor has instituted controls to insure that future milestones are met and is taking prudent action to regain some of the lost schedule.

   c. **Exceptional** - In the absence of excusable delays, contractor meets major milestone dates within the period within 10 working days. Contractor, as part of his management approach to the program, aggressively monitors all schedules and is able to anticipated counteract problems.

APPENDIX B

AWARD FEE REVIEW BOARD

The following members of OPM FATDS are preliminarily scheduled to evaluate the Award Fee Elements:

Table 7-2 AWARD FEE REVIEW BOARD

Award Fee Element			Evaluator
Chairman			PM FATDS, PM AFATDS
	Overall Technical and Programmatic Management		
		Overall Technical and Program Mgmt Ch. TMD	
	Technical		
		Optimum Utilization of CHS	AFATDS Lead Engineer
		Overall Logistics Management	Ch. RMD
		Implementation of Ada	Lead Software Engineer
		Embedded Training	Ch. RMD
		System Performance	AFATDS Lead Engineer
		Software Quality Process	Lead Software Engineer
		Software Quality	Lead Software Engineer
		Software Reusability	Lead Software Engineer
		Tests	APM Test
		MANPRINT	Ch. RMD
	Cost		
		Overall Program Cost	Ch. RMD
	Schedule		
		Contract Deliverables	PM AFATDS
		Program Schedule	PM AFATDS

A complete definition of these elements is provided in Appendix A.

Appendix C
Award Fee Evaluation Percentages

Table 7-3 Award Fee Evaluation Percentages

OVERALL TECHNICAL AND PROGRAM MANAGEMENT	5 MOS AC	SSR	SSR+ 6 MOS	CDR	CDR+ 6 MOS	SST	IOT&E
TECHNICAL	20%	20%	20%	20%	15%	5%	
Optimum Utilization of CHS	10%	5%		5%	10%		
Overall Logistics Management			5%	5%	10%	10%	
Implementation of Ada			10%	10%	5%		
Embedded Training					5%	10%	20%
System Performance							40%
Software Quality Process		10%	10%	10%	15%	20%	

Software Quality							40%
Software Reusability	10%	10%	10%	10%			
Tests					15%	25%	
MANPRINT	10%	10%	10%	10%	5%	5%	
COST	15%	20%	20%	15%	10%	10%	
SCHEDULE							
Contract Deliverables	15%	10%	5%	5%	5%	5%	
Program Schedule	20%	15%	10%	10%	5%	10%	

7.5.2 Contract Language - Award Fee Provision

(JIAWG Contract Elements for Software Reuse, 11 June 1990 DRAFT)
An award fee provision shall be included in the contract which rewards the contractor for outstanding performance in developing software components, and for effectively reusing such software components.

AWARD FEE CRITERIA
An evaluation shall be conducted on any components developed under this contract and delivered to the Government for inclusion in a Government software reuse library. The period for evaluation of award fee should be at least quarterly, and no longer than every six months. It is anticipated that the award fee pool will be allocated over the award fee periods with percentages applied that relate to development activity milestones. Award fee procedures should require the contractor to provide a self-assessment to the Government prior to each award fee review. The Government Evaluation Board shall include JIAWG representatives. The contractor's efforts shall be evaluated to determine an award fee based on the following:
Criteria for evaluating production of components:

1. Potential for reuse against other Government requirements
2. The degree to which the component is changeable
3. Degree of use of an information hiding structure
4. Quality of the reusable software object
5. Isolation of the classified portions of the components
6. Degree to which components exceed minimum requirements
7. Potential for cost savings, based on cost/benefit analysis

An award fee to evaluate accomplishments in reuse shall be considered based on the following criteria for evaluating use of components:

1. Degree that software objects are reused that exceeds what was proposed. This includes both components developed under this contract and originating from other sources.

2. Cost savings resulting from reuse, based on the actual cost of software compared with the proposed cost.

3. Improving upon proposed software schedule due to reuse, without negatively impacting the overall contract cost or schedule.

4. Demonstrated beneficial effect of reuse on productivity.

7.6 License Agreements

7.6.1 Example 1

- DFARS 252.227-7001 through 252.227-7012 provide examples of coverage for license agreements and royalty arrangements.

- Commercial licensing agreements abound. We have not included any here. DFARS 27.4 and the clauses identified there should be reviewed. Note that DFARS requires the full text of 252.227-7013 to be used when acquiring commercial software. Subsection (c) (1) (ii) addresses that software. Typically, many software vendors are intimidated by the clause; thus, the meaning may have to be explained.

- An example of a Cooperative Research and Development Agreement is included to show how these address license agreements.

7.6.2 Cooperative Research and Development Agreement

(AF Regulation 80-27 is one source for the sample agreement)

This agreement is between _______________ and the United States of America as represented by a Federal Laboratory of the United States.

It is a stated Congressional policy that the developments of Federal Laboratories be made available to state governments, local governments and private industry to stimulate the utilization of technology developed by the Federal Laboratories.

The Federal Technology Transfer Act of 1986 (P.L. 99-502) and Executive Order No. 12591, allow the Directors of Federal Laboratories to enter into Cooperative Research and Development Agreements with certain parties.

_____ has performed substantial research and development in modeling and simulation of _____. This research has generated data, know-how, and computer software. In particular, a computer program known as the _____ and its associated modules; _____ desires to transfer this technology to a third party to pursue further development of the technology and promote dissemination for the public good.

_____ is a DoD research, development and support company with demonstrated ability to refine and develop computer programs for use on various types of computer systems. _____ desires to further develop and market the technology.

The parties agree:

Article 1. Definitions

The following terms will have the definitions set forth below when used in this agreement. The definitions apply to the terms whether singular or plural.

1.1	Agreement means this Cooperative Research and Development Agreement
1.2	Invention is any invention or discovery (including computer software) which may be protected under Title 35 of the United States Code
1.3	The term "made" in relation to an invention means the conception or first actual reduction to practice.
1.4	Effective date is the date the last signature is affixed to this Agreement
1.5	Proprietary information is information which embodies trade secrets or which is confidential technical, business or financial information provided that such information: i) Is not generally known, or is not available without restriction from other sources; ii) has not been made available by the owners to others without restriction; iii) does not become publicly available; or iv) cannot be lawfully withheld from disclosure under the Freedom of Information Act, 5 USC 552.
1.6	The term created in relation to any copyrighted software means when the work is fixed in any tangible medium of expression for the first time, as provided for at 17 USC 101.
1.7	Technology means the computer program known as _____ and its associated modules, _____.
1.8	Software means software submitted to _____ by _____ for approval.

1.9 Government means the government of the United States of America.

Article 2. _____ Obligations

2.1 _____ will provide _____ a machine readable copy of the most current form of _____ source code on magnetic tape and associated written documentation. The material and information will be given to _____ within one month of the Agreement's effective date.

2.2 _____ will provide reasonable support to _____ throughout the term of this Agreement. In particular at _____ request, _____ will provide up to 5 model inputs with specified operations to be performed on the models. _____ will also provide _____ output for each model.

2.3 At _____ request, _____ will evaluate the results obtained by applying _____ generated software to the model. _____ generated software which passes _____ evaluation will be considered approved for _____ usage and distribution.

Article 3. _____ Obligations

3.1 _____ will assume the role of software configuration manager. Duties as configuration manager will include tailoring the program to meet the needs of commercial and Governmental users; approving and supervising the modification of the software by third party. Modification to meet a specific end user's needs may be performed by _____ under a contract with a specific user.

3.2 _____ will offer user support to the purchasers and users of the software including but not limited to: telephone consultation and a periodic newsletter for an annual fee.

3.3 _____ will take the appropriate steps to protect the intellectual property rights in the software.

3.4 _____ shall have the right to file patent applications on inventions made by _____ employees to the software. With respect to patents issued on such inventions, the government shall have a nonexclusive, nontransferable, irrevocable, paid up patent license to practice or have practiced for or on behalf of the Government the subject invention throughout the world.

3.5 _____ shall have the right to copyright those portions of any modified program to protect is modifications. With respect to programs so copyrighted, _____ grants the Government a non-exclusive, paid up, nontransferable copyright license throughout the world.

Article 4. Intellectual Property

4.1 _____ will pay _____ the sum of _____ for the software specified in Article 2.1. In addition, _____ shall pay a royalty of _____ of the sale price for each copy of the software sold and a royalty fee of _____ of the license fee for each license granted or renewed. _____ shall provide a quarterly accounting and pay _____ within thirty days of the end of each calendar quarter.

4.2 If, during the performance of this Agreement, _____ fails to make timely payments so _____, _____ may terminate this Agreement pursuant to paragraph 9.5

Article 5. Intellectual Property

5.1 This Agreement grants the Government no express or implied license rights under _____ owned background parent rights or copyrights. Any joint inventions made by employees and/or consultants of _____ and employees and/or consultants of _____, shall be owned jointly by _____, and _____ so that each party to this Agreement is licensed non-exclusively without the requirement of payment of any royalty to the other part.

5.2 _____ information. Information furnished _____ by _____ which relates to software developed solely at _____ expense, will be furnished with restricted rights as defined in DFARS 252.227-7013 and marked in accordance with the clause. Restricted rights software shall be used, reproduced and/or disclosed by _____ in accordance with the Government's rights in restricted software. In addition, _____ can use any version of the software internally royalty free.

Article 6. Representations and Warranties

6.1 Representations and Warranties of _____. _____ hereby represents and warrants to _____ as follows:

6.1.1 6.1.1 _____ is a command of the United States _____, which has as a substantial purpose the performance of research, development and/or engineering by its employees.

6.1.2 The performance of the activities specified by this Agreement are consistent with the mission of _____.

6.1.3 6.1.3 All prior reviews and approvals required by regulations or law have been obtained by _____ prior to the execution of this Agreement. The _____ official executing this Agreement has the requisite authority to do so. Notwithstanding the delegation of authority to execute this Agreement to the individual designated, the Secretary of the _____ has reserved to the Assistant Secretary of the _____, the opportunity provided by 15 USC 3710 a(c) (5) (A) to disapprove or require the modification of this Agreement within 30 days of the date it is presented to him or her by _____.

6.2 Representations and Warranties of _____. _____ hereby represents and warrants to _____ as follows:

6.2.1 6.2.1 Corporate Organization. _____ warrants it is a _____ corporation in good standing.

6.2.2 6.2.2 Power and Authority. _____ has the authority to enter into this Agreement.

6.2.3 6.2.3 Due Authorization. The Board of Directors and stockholder(s) of _____ have taken all actions required by law, Certificate of Articles of Incorporation or bylaws of _____ or otherwise, to authorize the execution and delivery of this Agreement.

6.2.4 6.2.4 No Violation. The execution and delivery of this Agreement, or an amended Agreement pursuant to paragraph 12.7, does not contravene any material provision of, or constitute a material default under any agreement binding on _____, or any valid order of a court of competent jurisdiction, regulatory agency or other body having authority to which _____ is subject.

6.2.5 In the event the Assistant Secretary of the _____ exercises the authority reserved in paragraph 6.1.3 above, _____ shall have 30 days from notification of any required modifications to either ratify the modifications or terminate this Agreement.

Article 7. *Termination*

7.1 Termination by Mutual Consent. _____ and _____ may elect to terminate this Agreement by Mutual Consent. In such event, the parties agree to specify the disposition of all property, patents and results of other work accomplished or in progress, arising from or performed under this Agreement.

7.2 Unilateral Termination

7.2.1 If _____ elects to terminate this Agreement for _____ failure to perform, then no further payment are owed to _____.

7.2.2 If _____ terminates this Agreement without cause and after certification of the software, all scheduled payments according to Article 4 are still due and payable on the dates noted to cover _____ costs incurred in evaluating and certifying the software.

7.2.3 7.2.3 Either party may terminate this Agreement after three years from the date of certification of the _____ software without cause.

Article 8. Capital Equipment

8.1 Capital Equipment. Any Capital Equipment furnished to _____ by _____ for testing _____ software shall remain the property of _____. Title shall remain in _____ and upon completion of testing, _____ shall bear the cost of removal, maintenance and return shipping to _____.

Article 9. Disputes

9.1 Settlement. _____ and _____ recognize that disputes arising under this Agreement are best resolved at the local working level by the parties directly involved. Both parties are encouraged to be imaginative in designing mechanisms and procedures to resolve disputes at this level. Any dispute arising under this Agreement which is not disposed of by agreement of the parties shall be submitted jointly to the head of the Agency or his designee for resolution.

9.2 Continuation of work. Pending the resolution of any dispute or claim pursuant to this Article, the parties agree that performance of all obligations shall be pursued diligently in accordance with the direction of the _____ signatory.

Article 10. Liability

10.1 Property. The U.S. Government shall not be responsible for damages to any property of ___ provided to __ or acquired by ___ pursuant to this Agreement.

10.2 Sponsor's Employees. __ agrees to indemnify and hold harmless the U.S. Government for any loss, claim, damage, or liability of any kind involving an employee of ___ arising in connection with this Agreement, except to the extent that such loss, claim, damage or liability arises from the negligence of ___ or its employees. The liability of ___ for such loss, claim, or damage shall be governed by Federal law.

10.3 No Warranty. ___ makes no express or implied warranty as to any matter whatsoever, including the conditions of the research or any invention or product, whether tangible or intangible, made, or developed under this Agreement, or the ownership, MERCHANTABILITY, or fitness for a particular purpose of the research or any invention product.

10.4 Indemnification. ___ holds the U.S. Government harmless and indemnifies the Government for all liabilities, demands, damages, expenses and losses arising out of the use by ___, or any party acting on its behalf or under its obligations, of ___ research and technical developments or out of any use, sale or other disposition by __, or others acting on its behalf or with its authorization, of products made by the use of ___ technical developments. This provision shall survive termination of this Agreement.

10.5 Force Majeure. Neither party shall be liable for any unforeseeable event beyond its reasonable control not caused by the fault or negligence of such party, which causes such party to be unable to perform its obligations under this Agreement and which it has been unable to overcome by the exercise of due diligence, including, but not limited to, flood, drought, earthquake, storm, fire, pestilence, lightning and other natural catastrophes, epidemic, war, riot, civic disturbance or disobedience, strikes, labor dispute, or failure, threat of failure, or sabotage of the __ facilities, or any order or injunction made by a court or public agency. In the event of the occurrence of such force majeure event, the party unable to perform shall promptly notify the other party. It shall further use its best efforts to resume performance as quickly as possible and shall suspend performance only for such period of time as is necessary as a result of the force majeure event.

Article 12. Miscellaneous

12.1 No Benefits. No member of, or delegate to the United States Congress, or resident commissioner, shall be admitted to any share or part of this Agreement, nor to any benefit that may arise therefrom: but this provision shall not be constructed to extend to this Agreement if made with a corporation for its general benefit.

12.2 Governing Law. The construction validity, performance and effect of this Agreement for all purposes shall be governed by the laws applicable to the Government of the United States.

12.3 Entire Agreement. This Agreement constitutes the entire agreement between the parties concerning the subject matter thereof and supersedes any prior understanding or written or oral agreement relative to said matter. Conflicting terms appearing in any attachment shall have no effect.

12.4 Headings. Titles and headings of the sections and subsections of this Agreement are for convenience of reference only. They are not a part of this Agreement and shall not affect its interpretation.

12.5 Waivers. None of the provisions of this Agreement shall be considered waived by any party unless such waiver is given in writing to all other parties. The failure of any party to insist upon strict performance of any of the terms and conditions, or failure or delay to exercise any rights provided by this Agreement or by law, shall not be deemed a waiver of any rights.

12.6 Severability. The illegality or invalidity of any provisions of this Agreement shall not impair, affect or invalidate the other provisions of the Agreement.

12.7 Amendments. If either party describes a modification to this Agreement, the parties shall, upon reasonable notice of the proposed modification by the party desiring the change, confer to determine the desirability of such modification. No modification shall be effective until a written amendment setting forth the modification is signed by all parties' authorized representatives.

12.8 Assignment. Neither this Agreement nor any rights or obligations of any party hereunder shall be assigned or otherwise transferred by either party without the prior written consent of the other party. Except that, __ may assign this Agreement to the successors or assignees of a substantial portion of ___ business interests to which this Agreement directly pertains.

12.9 Notices. All notices pertaining to or required by this Agreement shall be in writing, signed by an authorized representative, and either delivered by hand or mailed, postage prepaid, addresses as follows:

If to ______: (point of contact name and address)

If to ______: (point of contact name and address)

Any party may change the address by giving notice to the other parties at the address above.

12.10 Independent Contractors. The relationship of the parties to this Agreement is that of independent contractors and not as agents of each other or as joint venturers or partners. ____ shall maintain sole and executive control over its personnel and operations.

12.11 Use of Name or Endorsements.

(a) ____ shall not use the name of ____ on any product or service which has not been approved and evaluated under paragraph 2.3 without the prior written approval of ____.

(b) By entering into this Agreement, ____ does not directly or indirectly endorse any product service provided, or to be provided by ___, its successors, assignees, or licensees. ___ shall not in any way imply that this Agreement is an endorsement of any such product or service.

Article 13. Duration of Agreement and Effective Data

13.1 13.1 Duration of Agreement. It is mutually recognized that the development program cannot be rigidly defined in advance, and that good faith guidelines, subject to adjustment by mutual agreement, to fit circumstances as the development program proceeds. In no case will this Agreement extend beyond three from the date of this Agreement, unless it is revised in accordance with Article 12 of this Agreement.

13.2 Effective Date

This must be submitted to the Office of the Assistance secretary of the_____ at (give mailing address) for review. Receipt of this document by ____ will begin a THIRTY (30) day period during which the Agreement may be disapproved or modification required.

This must be submitted to the Office of the Assistance secretary of the_____ at (give mailing address) for review. Receipt of this document by ____ will begin a THIRTY (30) day period during which the Agreement may be disapproved or modification required.

IN WITNESS WHEREOF, the parties have caused this Agreement to be executed by their duly authorized representative as follows:

FOR _______

(name and signature)

On this (date), before me appeared (person's name), the (title) of (company), who I hereby certify signed this instrument on behalf of ____ and in so doing was acting within the scope of his authority and has bound that organization. In witness whereof I have signed below

(name and signature)

FOR:

(name and signature)

DATE

THIS DOCUMENT IS HEREBY SUBMITTED FOR REVIEW AS REQUIRED BY THE POLICY SET FORTH IN THE ABOVE PARAGRAPH. IF NO NOTICE OF DISAPPROVAL OR REQUIRED MODIFICATION IS RECEIVED FROM THE REVIEWING AUTHORITY PRIOR TO (Date), THIS Agreement SHALL ENTER INTO FORCE AS OF THE DATE OF THE SIGNATURE OF THE REPRESENTATIVE OF _____, WHO WILL BE THE LAST TO SIGN.

SUBMITTED FOR REVIEW (date).

(name and signature)

7.7 Copyright Assignments (Contract Section H)

7.7.1 Example 1

When assignment of copyright is deemed necessary to protect derivative works, or for other reasons, a clause similar to the following may be used. Note: assignment for Government purposes is automatically obtained under DFARS 252.227-7013. Total assignment would remove the contractor's commercial position protection.

> The contractor agrees to provide total assignment of copyright for the following (software/data) to the Government, and hereby revokes the right to any future claims of copyright protection.
>
> (Identify Products)
>
> The contractor recognizes and agrees that this assignment allows the Government complete control of the identified products, including any products derived from them which are produced under Government funding.

7.7.2 Example 2

Rights in Technical Data and Computer Software/Copyrights.
The contractor shall provide copyright releases, as defined in paragraph 3.3.4 of MIL-M-7298D. The contractor shall comply with Department of Defense Federal Acquisition Regulation Supplement (DFARS) clause 252.227-7013, "Rights in Technical Data and Computer Software (Oct 1988)" with respect to technical data and computer software generated and/or furnished under the contract. Upon completion of the contract, the contractor shall deliver to the Government the entire database, including unique programs and operating instructions.

7.7.3 Example 3

TRANSFER OF COPYRIGHT AND RESERVATION

AND LICENSE OF EXCLUSIVE RIGHTS

FOR COMMERCIAL PURPOSES

_____ is the rightful owner of the copyright of the computer software and computer software documentation listed in Schedule I, and hereinafter collectively referred to as _____ Software and Documentation which was developed and created by _____ under contracts with the United States Government.

The United States Government as represented by the Secretary of the Army ("Government") desires to own the copyright in the _____ Software and Documentation and _____ desires an exclusive worldwide license in the _____ Software and Documentation for all commercial purposes.

Subject to the reservations and licenses set forth below, _____ hereby transfers, assigns and conveys its copyright interest in the _____ Software and Documentation to the Government to hold for the duration of the copyright.

_____ hereby reserves, and the Government grants to _____ an irrevocable, transferable, fully paid-up, royalty fee, worldwide exclusive right and license to use, sell, lease, license, reproduce, prepare derivative works of, and distribute the _____ Software and Documentation for all purposes other than transactions with the Government.

_____ hereby reserves, and the Government grants to _____ an irrevocable, transferable, assignable, fully paid-up, royalty fee, non-exclusive right and license to use, reproduce, prepare derivative works of, and distribute the _____ Software and Documentation for the Government.

The Government shall promptly provide to _____ copies of all derivative works and upgrades to the _____ Software and Documentation and works related thereto. Such derivative works, upgrades and related works shall be subject to the same reservations and licenses set forth above.

Schedule - Software and Documentation

Description of Computer Software:

- Distributed Computing Design Systems (DCDS), (specify).
- Distributed Computing Design Systems (DCDS), (specify).

Description of Computer Software Documentation:

1. Distributed Computing Design System (DCDS) Tools User's Guide, (specify).
2. Distributed Computing Design System (DCDS) Methodology Guide, (specify).
3. Distributed Computing Design System (DCDS) Maintenance Manual, (specify).
4. Distributed Computing Design System (DCDS) Technical Overview, (specify).

U.S. GOVERNMENT	(SOFTWARE DEVELOPER)
_______________	_______________
Name	Name
_______________	_______________
Title	Title

_______________ _______________

Date Date

Certificate
I,_____, certify that I am the Secretary of _____ and that (name of signator) who signed this document on behalf of _____ was then the (signatory's title); that this document was duly signed for and in behalf of _____ by authority of its government body within the scope of its corporate powers.

(Corporate Seal) ____________________

7.8 Notification of Unlimited Rights/Less Than Unlimited Rights

7.8.1 See DFARS 252,227-7019

7.9 SOW Language for Reuse

From the Statement of Work for the F-22/NATF, Advanced Tactical Fighter Weapon Systems

The Contractor shall design, analyze, develop, establish interfaces with the airframe test, integrate, fabricate, and qualify the Avionics Segment to satisfy the requirements of the F-22 Avionics System Segment Specification (52R5009) in accordance with the Avionics Segment Integrated Master Plan and the Avionics Integrity Program Master Plan. The Contractor shall participate in and contribute to activities to promote systems commonality between the Air Vehicle, Training System, and Support System.

The Contractor shall participate in and contribute to JIAWG activities to promote systems commonality at the module level between F-22 and Army Light Helicopter programs. The Contractor shall support activities to complete and validate the Common Avionics Baseline Specifications, including regular status reporting. The Contractor shall, in collaboration with the LH prime contractor, plan, support, and execute an F-22/LH Common Module Exchangeability Demonstration. This demonstration shall verify that commonality at the module level for equivalent form/fit/function/interface (F3I) has been achieved between the avionics suited of the two aircraft.

The Contractor shall provide a Flying Test Bed for integration test activities for the FGSD avionics system. The Contractor shall also utilize the F-22 S/SEE for software development and develop ground based laboratory facilities, including a System Integration Laboratory to: support buildup of avionics systems and integration with other aircraft systems; validation/verification of avionics functions and interfaces; perform avionics/VMS integration and functional validation tests; support flight test and anomaly testing, and accommodate testing of identified growth./ advanced technology items. Reporting requirements shall be in accordance with the CDRL (DI-NDTI- 80809/T, OT- 90-34210, OT-90-34212, DI-MISC-80296/T, OT-90-34222).

The Contractor shall maximize the practical level of commonality between the F-22 and NF-22 avionics. Specifically, the Contractor shall provide comparison detailed system, subsystem and module level physical and electromagnetic environments. The Contractor shall participate in the Air Force/Navy team comparison review by identifying performance, cost, schedule, and implementation impacts to each service based on operational and environmental requirements derived from the AVIP process and stated Navy goals.

The Contractor shall address the maximum feasible use of common hardware and software computer resources across the F-22 weapon system and with JIAWG standards. The use of common elements shall be identified where justified on the grounds of cost, performance, and supportability, and they can be developed to conform to the applicable requirements of the F-22 System Specification.

APPENDIX A - CURRENT GOVERNMENT SOFTWARE REUSE PROGRAMS

The following table shows some Government reuse programs and their focus and status as it is known today. Readers are encouraged to contact the organizations responsible for the programs to gain a greater appreciation and benefit of their lessons learned.

Table A-1 Government Software Reuse Programs

Program	Reuse Approach	Organization	Comments
ASSETS	• Distributed network of reuse libraries • On-line STARS and horizontal domain library • National Software Reuse Directory	ARPA (STARS)	• Operational library • Interconnection with CARDS since 10/92
1. ATCCS CHS ASAS	• CHS will develop reusable components for ATCCS use • ASAS requiring use of existing software	U.S.Army Communications-Electronics Command (PEO CCSD/PM CHS)	
JAST	• Joint Program Office defines advanced suite of electronics and armaments systems to be used on next generation attack aircraft	Air Force, Navy (NAVSEA), Marine Corps	• Analysis and design effort for R&D • Analysis and design effort for R&D
CARDS	• Developing and transsitioning reuse techniques and technologies • Developing and trainsitioning related training materials • Domain-specific library development	U.S. Air Force A.F. Materiel Command/Electronic Systems Center (ESC/ENS)	• Phase I, Proof-of-Concept, completed • Phase II, operational library, Feb 92 • Phase II, operational library, Feb 92 • Phase IV, Franchise implementation, FY 94
DSRS	• Strong emphasis on domain and reuse engineering • Certified components faceted classification scheme	Defense Information Systems Agency (DISA) (DISA/CIM/XER)	• Library operational with extractable components and pointer to commercial software • DSRS supports remote software reuse sites

FEDERAL SOFTWARE EXCHANGE CENTER	• Sharing of common-use software and related documentation DoDI 7930.2 applies Federal Information Resources Management Regulation (FIRMR) (Sub Part 201-24.2) applies FIRMR Bulletin C-12 applies	Department of Commerce National Technical Information Service	• DODI 7930.2 defines scope and applicability • NTIDS maintains current catalogue • No proprietary nor classified software
GPALS	• Software Reuse Strategy Plan developed for use by all GPALS segments	U.S. Army Strategic Defense Command (SDIO) (CSSD-CR-S)	
Piece Shield	• Contractor(Hughes) reusing air defense software components developed in other in-house programs	U.S. Air Force A.F. Materiel Command/Electronic Systems Center (ESC/AVS)	• Software PDR conducted July 92 • Approach successful to date
PRISM	• Develop reusable C2 software components • Populate CARDS library	U.S. Air Force A.F. Materiel Command/Electronic Systems Center (ESC/ENS)	• Provides Commnad Center Component • Rapid prototyping facility

APPENDIX B - CONTRACT CONSIDERATIONS FOR SOFTWARE REUSE

(From the Reuse Acquisition Action Team (RAAT), ACM SIGAda Reuse Working Group)

A checklist consisting of contractual topics that should be considered prior to the Request For Proposal (RFP) process was developed by the Pre-RFP and Source Selection Working Group during the DoD Systems Acquisition and Software Reuse Workshop [17]. This workshop was held 5-7 November 1991 and was sponsored by the Special Interest Group (SIGAda) Reuse Working Group and the Institute for Defense Analysis. The purpose of this workshop was to determine the next step in initiating and encouraging software reuse in the DoD by identifying critical issues and recommending solutions to problems in the following areas: Pre-RFP and source selection, requirements definition, incentives and inhibitors, measures and cost analysis, and roles of government and industry. Working groups were established to address the problems in each area.

The purpose of the Pre-RFP working Group was to determine how and what to incorporate into the procurement process to encourage reuse. One outcome of this panel was a checklist, which is directed towards both technical and acquisition strategy personnel. The purpose of this checklist is for planning to incorporate reuse at the initiation of a project, and prior to each acquisition rather that for project evaluation. It also can be used by those who review and approve program and acquisition strategies and plans. The checklist applies to two categories of reuse: developing products for reuse and reusing previously developed products. Use of this checklist will ensure a consistent approach to incorporation of reuse in all parts of acquisition documentation and requirements. The checklist considerations should be applied to all acquisitions which require software development or modification. The checklist provides a structured approach to identify reuse potential and promote reuse.

This list is actually a composite of four lists:

1. contractual topics to consider prior to the RFP process;
2. data rights issues;
3. proposal evaluation and
4. cost evaluation.

The reuse topics to consider at the initiation of a project and prior to each acquisition include: cost/benefit trade-offs; domain analysis; library requirements' source of components; reuse plan and warranty.

The purpose of the data rights topics are so that the Government is able to decide early on the reuse requirements and what it needs to fulfill the mission strategy. The Government must also make certain that the correct data rights clauses are contained in the contract. Some issues addressed in the proposal evaluation checklist are: determine need for an on-site reuse assessment; have contractor demonstrate reuse expertise, technical approach, reuse management plans, and organization structure. The cost evaluation checklist encourages contracting personnel to define the cost method that will be used (life-cycle, program development, and domain components) and to specify data to be submitted with the cost proposal.

Checklist of Data Input Required From Technical and Acquisition Strategy Personnel Prior to RFP

Should reuse efforts be included in this contract or in a separate contract (parallel effort)?

Choose whatever type of contract is most appropriate for the procurement. In order to encourage software reuse, including development of reusable software, consider using an Award Fee provision in the contract. In this case, an Award Fee Plan must be developed. See JIAWG Contract Elements or AFATDS Award Fee Plan for examples of evaluation factors and criteria.

Is reuse required? Is the requirement for reuse of previously created reusable components, or is it for development of reusable components under this contract or is it both? Consider whether the current software development is unprecedented? If reuse is required, provide input on Statement of Work requirements. (Contract section C)

Is reuse not required but encouraged? If so, provide input to contractor to convey this goal. (Contract section C)

Will the contractor be required to make trade-offs between degree of reuse and technical performance, cost, and schedule? If so, provide some guidance on what the goals are. (Contract section C)

Is domain analysis required by the contract? Has it already been done? If not, who will do it? Are cost and scheduling of domain analysis included in the government's estimates? If domain analysis is either required or provided to the contractor, what are the specific tasks, reference documents, and expected outcome? (Contract section C)

Are there any reuse deliverables specifically required? Are these listed in the contract? If so, have the contractor identify separate costs required to make items reusable (cost of value added for reuse). (Contract sections B, F)

If there are separate deliverables of reusable components (e.g., to a central library), what data, documents, test tools and results, quality tools and analysis, design trade-off analysis, etc. are required with the component? (Contract Data Requirements List–DD Form 1423)

Is there a requirement for a library? Is it internal or external? If external, what procedures, processes, etc., does the library require to access data or to provide data? (Contract sections C, H)

Government should identify sources where the contractor can obtain software components that may have reused potential. Identification and location of specific candidate components can be listed, if known. (Contract section H)

Should the contractor be required to write a reuse plan or an explanation of internal reuse policies? Is this a separate document or is it part of the software development plan (SDP)? (Contract section C, DD Form 1423)

Contractor Warranty Requirements: Choose one or more DFARS warranty provisions appropriate to the reuse requirements of the contract (development for future reuse, development with reuse "as is," development with reuse-modified), and include in the contract. (Contract section H)

Government Warranty for GFE: If any software is to be provided as GFE, address government warranty/liability if not covered by GFE clause(s). (Contract section H)

Data Rights Issues

Government must decide up front, based on future intended reuse requirement, what rights it requires, consistent with a strategy to encourage reuse. Consider whether allowing the contractor to retain some rights would encourage greater reuse.

Assure that data rights clauses are in the contract and that any software components to be delivered with restricted rights are identified in the proposal as provided by DFARS 252.227-7013 and 252.227-7019). (Contract section I,L)

DFARS 252.227-7026, dealing with deferred delivery of technical data or computer software, and DFARS 252.227-7027, dealing with deferred ordering of technical data or computer software, apply to this contract and may be specifically applied to components (Contract section I)

Checklist for Proposal Evaluation

Source selection decision (competitive or sole-source)?

If competitive, what are the evaluation criteria? What are the particular criteria for reuse? Provide reuse input to the Source Selection Plan. (Contract section M)

Is there a need for an on-site assessment to evaluate reuse capabilities? (Contract section M)

Will a sample problem enhance evaluation? If so, provide text and evaluation criteria. (Contract section M)

Have the contractor demonstrate reuse expertise, reuse technical approach, reuse management and organization. (Contract sections M, L)

How will you evaluate the risk impact of reuse (positive or negative as proposed); for example, assess risk by applying a percent factor times the proposed lines of code using the three categories: reuse "as is," reuse-modified, no reuse.

Identify what data the Contractor is to submit with the proposal for evaluation (e.g., draft SDP specifically addressing reuse). (Contract section L).

Cost Evaluation

How do you evaluate the cost of reuse (life-cycle cost, program development cost, domain component cost)? Define the methodology. (Contract section M)

Consider adjusting a contractor's proposed cost for evaluation based on the contractor's proposed approach to reuse by applying weighting factors based on the quality and quantity of reuse. For example, evaluation cost can be increased if there is no reuse and decreased if there is extensive reuse. (Contract section M)

Reference List

1. Report of the JLC, San Antonio I - Software Reusability Panel (Panel IV), 28 January - 1 February 1991.

2. NSIA Study, "The Business Issues Associated with Software Reuse," 15 December 1990.

3. "US45 - Current FAR and Budget/Finance Environments", STARS-SC-03501/001/00 and STARS-SC-03504/001/00, 30 March 1990.

4. "Advanced Field Artillery Tactical Data System (AFATDS) Award Fee Determination Plan", 30 Mar 91.

5. "Contract Elements for Software Reuse" (Draft), Prepared by (JIAWG) Software Task Group, 13 June 90.

6. "Strategy and Mechanisms for Encouraging Reuse in the Acquisition of Strategic Defense Initiative Software", IDA Paper P-2494, June 1990.

APPENDIX C - A CASE STUDY IN SOFTWARE REUSE

Introduction

The following case study shows how the issues in the handbook are applied to an acquisition to determine whether and how to incorporate software reuse. The case addresses the following reuse acquisition process steps: technical planning, cost, contracts, software rights, risks, incentives, evaluation criteria, and evaluation standards.

Scenario

The Government has a requirement for an intelligence system trainer. The trainer will be used to assist operators in gaining proficiency prior to sitting at stations and actually working with live intelligence data. It is a general purpose training system to provide students an initial exposure to the intelligence field.

The Government has already attempted to acquire this trainer system under a prior cost reimbursement contract. The work was not fully completed. The contractor delivered:

Table C-1 Contractor Deliveries

• Specifications	• Software
• Training Manuals	• Hardware

All software (except COTS) was delivered with unlimited rights

The delivered products do not meet all the Governments requirements. The Government has concluded that the contractor was on the right track but overwhelmed by the task, ultimately delivering products ranging from adequate to poor quality. Some of the more significant status issues are:

- Software is hardware dependent (operable on single CPU)
- Data Base Management System (DBMS) was uniquely developed by the contractor and is not well suited to its task.
- Software architecture is clumsy with too many CSCI's (Although some CSCI's, including code, are adequate).
- Software was developed in Ada using current accepted practices.

The user's requirement remains valid. The PEO responsible for satisfying the requirement has tasked the Program Manager to deliver a trainer system capability as soon as possible. There are limited funds available (albeit, probably sufficient) since this "new" acquisition was not planned in the period it will be executed. The PEO has not decided what software strategy to pursue - whether to encourage new software development, reuse existing software or some combination. Some pertinent considerations he faces include:

- The Government definitely wants to pursue an Open System Architecture (OSA) for the software.
- There have been a number of new COTS software products introduced in the DBMS and secure software areas.
- The PEO believes there are GOTS products which may be potentially used in this program.
- The PEO believes, though he has no validated proof, some products from the prior contract should be salvageable.
- There are future training systems planned which will have similar capabilities/functionalities.

Objectives for Solution

- Require domain analysis to identify and model the existing domain to define a core architecture from which existing components (COTS/GOTS) can be derived to meet architectural requirements.
- Conduct cost/benefit analysis to assess new development versus reuse.
- Reuse existing software components from prior contract (either "as is" or modified).
- Determine impact of all new software development. Would it: take too long; be more expensive; be riskier; or be reusable without a reuse strategy?
- Provide solution of:

  Some new software developed for reuse,

  Reuse existing software from old contract (as is or modified),

  Use of COTS/GOTS.
- Discuss:

  Integration risks; what should be tested; when and how,

  Software rights.
- Provide support to Program Manager.

  Assess Training needs.

  Evaluate the need and extent of incentives (both Government and contractors).

Analysis/Solution

An initial survey and preliminary program analysis is conducted to uncover all pertinent domain information to develop a reuse strategy (See Section 2.4.1, Program Technical Baseline). All programs and information in the domain should be included in the analysis.

Data collected during this preliminary program analysis should establish the context of the domain in relation to related horizontal and vertical domains as well uncover answers to the following questions:

- What portions of the prior system can still be used?
- What are the requirements for systems to be developed in the near-future?
- What components from this system development could be used on these future systems?
- What existing components (COTS or GOTS) could be used?
- What other systems/components exist that meet the system's requirements/specifications (i.e., NDI)?
- What resources are required?
- What resources are available

In analyzing the status of the domain, Table, 2-2, Reuse Technical Baseline can be utilized (only applicable items from this table are discussed). First, the requirements are examined to determine needed components and resources. Next, the environment is analyzed to determine what components exist, the nature of the domain and the available resources. Then, what is needed is matched to what is available.

Domain Boundary/Domain Status (General)

Is there an existing domain within which the system belongs?

Yes, training systems.

Is the domain mature, understood, and stable?

The domain of training systems is mature and understood. Training systems have been evolving over the years, but many exist and the concepts are well understood.

Is the technology stable and predictable?

Although many training systems include new/recent technologies, such as CD-ROM, hypertext and hypermedia, they have not replaced some of the older technologies, such as laser-disk technology, completely. All of these technologies used in training systems still remain in the repertoire to choose from.

Is there a demand for these type of components?

User interface, architecture, domain model and perhaps some database requirements can be used by other training systems.

What is the supply of components?

The user interface and data management algorithms can be reused.

Component Requirements (Needs)

What types of components will be needed?

The following components are needed for the system: domain model, software architecture, system design, user interface, DBMS, message handler, and data management techniques.

What descriptors are needed?

If COTS products are used, need to know: vendor, cost, license availability and cost, and hardware specifications.

If GOTS are used, need to know: developer, maintainer, and hardware specifications.

Component Analysis (Availability)

Are there components available (other than from libraries) that meet requirements?

Yes, more than one commercial DBMS is on the market which meet the requirements. The user interface from the prior system development can be reused

What components are needed, are not available, but can be reused by others (i.e. develop reusable components)?

Domain model, software architecture, system design.

What components are needed and are available (i.e., reuse existing components)?

User interface (as is), DBMS (COTS, as is), message handler (needs modification), and data management techniques (needs modification).

Who has ownership?

COTS: DBMS

GOTS: user interface, message handler, and data management techniques

What are costs to obtain, modify, test, integrate, and certify?

Libraries

Are there libraries which cater to this domain?

No, there is a lack of libraries for this domain.

Technical Resources

What types of personnel are available?

The assigned program Office has experts in the domain (i.e., intelligence training experts).

There are currently no available domain engineering experts (i.e., those familiar with: domain analysis, domain modeling, developing systems using existing components, and developing components to be reusable).

What types of metrics are available?

No metrics are available (for costing and level of reuse achieved).

The preliminary analysis of the program in its domain has also uncovered the following risks (mitigations for these risks are discussed in the text below):

- Lack of knowledge about domain analysis concepts (lack of domain engineering experts) (See Table 2-4, Technical Risks and Mitigations: Requirements)
- Lack of domain models (See Table 2-4, Technical Risks and Mitigations: Requirements)
- Lack of libraries and software reuse tools (See Table 2-5, Technical Risks and Mitigations: Software Development)
- Integration risks; what should be tested; when and how (See Table 2-7, Technical Risks and Mitigations: Performance)?
- Lack of metrics availability for costing and level of reuse achieved (See Table 2-8, Cost/Schedule Risks and Mitigations)
- Possible impact to programs' schedule and cost, due to other risks (See Table 2-8, Cost/Schedule Risks and Mitigations)

The above analysis results in the following. There is a fairly high commonality across intelligence training systems and other domains that also use the common features of training/simulation systems. The primary features of the system to be compared to other systems are: message handling, data management (for both classified and unclassified data) and the user interface and software architecture (See Table C-2, Common Functionalities Across Programs).

Table C-2 Common Functionalities Across Programs

FUNCTION	PROGRAMS			
	Current Program	Program #2	Program #3	Program #4
Message Handling	High Commonality	Good Commonality	High Commonality	High Commonality
Data Management	Good Commonality	Good Commonality	Good Commonality	Good Commonality
DBMS	Good Commonality	Good Commonality	Good Commonality	Good Commonality
Software Architecture	High Commonality	Good Commonality	Good Commonality	High Commonality
User Interface	High Commonality	Good Commonality	High Commonality	Some Commonality

The prior system was developed using currently accepted Ada program practices and the preliminary analysis shows that the user interface is acceptable and can be salvageable. Both classified and non-classified data management algorithms are needed for the system. More than one commercial DBMS meets the requirements of the system and may be used. The software architecture and perhaps even the user interface could be reused on future training systems that have similar capabilities/functionalities.

Although the Program Office has access to experts in the application, they are lacking personnel who have domain engineering expertise. An outside organization, another government agency or an independent contractor will need to be called in to at least conduct the in-depth domain analysis and to capture the domain knowledge so that it can be reused on future projects. As an example, an outside organization which has software reuse experience, such as the Software Engineering Institute (SEI), would be a good choice, since they would have expertise in both training systems and domain engineering practices. The Program Office could assist the outside organization by providing a few personnel, both acquisition managers and engineers, from the Program Office, and perhaps from other programs, who have experience in the application area. This would not only reduce manpower required from the outside organization and provide them with domain experts, but also provide valuable on-the-job training in software reuse practices to the Program Office personnel. In turn, these people could bring back the domain engineering knowledge gained from the effort and utilize it on the development phase. Perhaps, these people could, in the future, be offered to work with other reuse programs in the domain or even be a guest speaker at the various acquisition courses offered within DoD. In fact, these additional duties could be used as an incentive to promote reuse.

Having the outside organization conduct the in-depth domain analysis, including developing the domain model, will help reduce some of the risks. Some of the risks associated with the lack of proposed metrics definitions and collection procedures can be reduced, since they have been established by the SEI and can be used on the project.

The Program Manager must not be penalized for the increase in schedule and increased cost for incorporating reuse into the program. The funding for the domain analysis would come from R&D dollars and be expensed against several programs since other programs will be able to use some of the components developed under this effort in the future.

It is decided to have two independent cost estimates prepared: one, under the Program Managers responsibility, for the reuse alternative; one, by an independent contractor, for a new software development solution. These will be accomplished by quantitative risk assessments. The cost estimates and risk assessments will be available for the business strategy panel which will be chaired by the PEO.

The Business Strategy Panel, which included user representation, considered the following information on the reuse versus new development alternatives. The reuse cost and risk assessment takes advantage of the domain analysis which was performed (See Sections 2.4.3 and 7.4). A description of the variables from Section 2.4.3 is repeated here:

Scn_1 =	Software development costs of new software not developed for reuse (entire system)
Scn_2	Software development costs of new software not developed for reuse (portion of entire system)
Scr =	Software development costs for reusable software
Scm =	Software development costs for software modified to be reusable
Scs_1 =	Software used "as is" (NDI, GOTS, COTS) for new development without planned reuse
Scs_2 =	Software used "as is" (NDI, GOTS, COTS) for reuse approach
Sco_1 =	Operation and maintenance costs for software not employing reuse
Sco_2 =	Operation and maintenance costs for software employing reuse

Table C-3 Case Cost Estimates

	In Millions of Dollars	
	Reuse	**New Development**
Software Development costs	$2.8	$2.6
(includes anticipated license costs for reuse software and/or COTS	(Scr+Scm+Scn_2+Sci)	(Scn_1+Scs)
Operations and Maintenance Costs (10 year life cycle)	$3.0 ($Sco_2$)	$3.6 ($Sco_1$)
Total	$5.8	$6.2

Risks for reuse have been identified in the areas of software integration test and documentation since there is little experience in the impact of reuse on these areas. The Program Manager also identifies schedule risk because of limited experience. The Program Manager concludes that cost variance potential for these risks could cause up to a 20% increase in development costs ($.56M = .2 * $2.8M) and up to 5% increase in O&M costs ($.15M = .05 * $3M). He assesses low risk in O&M since he anticipates a lesser requirement for training materials and increased proficiency based on use of existing software.

The independent contractor assesses risk in software development, test and integration, leading to a 15% potential increase ($.39M = .15 * $2.6M). O&M has a 10% variance ($.36M = .1 * $3.6M) potential for anticipated increases in post deployment modifications needed if development does not adequately consider software maintainability design implications.

The adjusted cost estimates become:

Table C-4 Case Adjusted Cost Estimates

	Reuse	New Development
Development and O&M	$5.80	$6.20
Risk	$0.71	$0.75
Total	$6.51	$6.95

It is also noted that the reuse cost estimate also identified commonality in future systems components development in other system within the domain which will achieve at least 20% development cost avoidance on four programs (estimated savings of $6M).

Based on this information and the knowledge that future savings will occur on other programs, the PEO decides to mandate a reuse solution. The PEO also notes that the current program will serve as the lead program for the reuse strategy and will be provided additional funds (See Table C-5, Reuse Cost Apportionment) to develop reusable components; validate existing components and provide higher quality documentation. The PEO also notes the future programs (Programs 2, 3, and 4 in Table C-2) will require extra funding for distinct reuse integration, test and documentation. However, these will be offset by savings from reuse so these future programs will not require additional funding. The current programs will also receive funds for interface with the library which will house the reusable components.

Table C-5 Reuse Cost Apportionment

PROGRAMS IMPLEMENTING REUSE	REUSE COST APPORTIONMENT			
	Single Program	Multiple Programs	Lead Single Program	New Separate Reuse Program
Current Program	•	•	•	
Program 2, 3, 4		•		

The PEO instructs that the RFP evaluation criteria address reuse considerations, such as (1) software reuse process methodology; (2) domain analysis; (3) software metrics; (4) reuse management; and (5) software rights. The Government will pursue a contractor O&M approach for the first two years of system deployment. It will not consider calling for election of any unlimited rights to which it may be entitled until the second year of contractor O&M (See Table C-6, Reuse Environments and Ownership). The Program Manager will accomplish this by deferring delivery of data which supports the full software architecture. Since development and deployment are expected to be completed four years after award, this could allow the contractor some inherent competitive advantage in other Government programs and preserve its commercial exclusivity until rights are provided. Of course, this incentive is good for new software development, but the PEO is anticipating that industry will reuse some of its own non-developmental software, which is not yet commercialized, if it knows this incentive exists. The Government will entertain any innovative approach to software ownership issues which preserves its ability to complete the last eight years of O&M in a cost effective manner. This includes never claiming full unlimited rights if the best interest of the Government are preserved for future programs.

Table C-6 Reuse Environments and Ownership

Reuse Environments and Ownership				
Type of reuse	**Anticipated frequency of reuse**	**Anticipated reuse environment**	**What should be owned**	**How should components be owned**
• **New development for reuse** • **Modify existing component** • Use "As Is" GOTS COTS NDI	• Single program • **Multiple, specific programs** • **Multiple, specific programs** • *Unlimited government programs application* • **Multiple, specific programs** • One-time use • Multiple, but limited, non-specific programs • Unlimited use	• **Development** • **Test** • **Support** • **Post-Deployment Software Support** • *Modifications/ upgrade* • *Derivative Products*	• **Architecture** • **Detailed design** • **Code** • **Supporting documentation** • **Training Manuals**	• **Licensed** • Unlimited rights • Restricted rights • Restricted copyright • **Deferred delivery of rights** • Deferred ordering of rights • Escrow for "Fail Safe" purposes • Phasing out of government rights over time

Legend: Bold = Applicable to this effort Italic = Undetermined at this time

The Business Strategy Panel also asks that specific milestones be included in the contract to assess progress in developing reusable software. Additionally, metrics are identified and/or developed to help assess schedule, technical and costs progress in the reuse area. The user requests, and all agree, that database generation for reuse peculiar aspects of the program be required so it can be available to develop reuse effectiveness measurements for this and other programs.

Implementation

The Program Manager notes that help would be needed to develop evaluation criteria, proposal instruction and contract language to address all the issues discussed. A team of technical, contracts, financial and legal personnel is formed to assist in this effort. The Business Strategy Panel identifies several sources of templates to help in this area, most prominently, the Acquisition Handbook as well as other DoD programs where reuse is being pursued.

Subsequent to the Business Strategy Panel, the following actions were accomplished by the Program Manager.

Cost. (Subpart to the Business Strategy Panel) The Program Manager revalidated the cost estimate for the reuse approaches. There was no variation from the initial estimate.

Contracts.The program involves development of reusable software, in addition to probable modifications of existing software to make it reusable. Coupling these factors with risks previously identified, and the Governments requirement to work closely with the contractor results in a decision to use a cost reimbursement type contract (See Section 2.4.5, Contract Types, Figure 2-4, Program Risk and Contract Type, Figure 2-5, Government Involvement and Contract Type, and Figure 2-6, Extent of Reuse and Contract Type).

Since this is a new contract, recoupment is not applicable for other than foreign military sales.

Rights.The PEO asked the Program Manager to revalidate conclusions regarding rights strategies. This was done using Table C-7, Government/Contractor Reuse Objectives (Table 2-17) and Table C-8, Anticipated Reuse Environment Matrix (Table 2-14) (See Section 2.5.1, Software Rights).

Table C-7, Government/Contractor Reuse Objectives, was used to understand Government and contractor objectives and to identify non-conflicting intersections. These intersections help identify the most appropriate incentives for reuse. The matrix is used for both Government and commercial markets to establish potential compatibilities/incompatibilities for each. (For each cell in the table, the text above the diagonal line applies to Government objectives and text below the diagonal applies to contractor objectives.)

Table C-8 indicates that reuse in this efforts applies to row 2, "Multiple, specific program applications" only. Thus the following rights discussion pertains to multiple, specific program applications.

Table C-7 Government/Contractor Reuse Objectives

government	GOVERNMENT - CONTRACTOR OBJECTIVE			
contractor	Single Program Reuse	Multiple/Specific Programs Reuse	Unlimited Programs Reuse	Public Disclosure
Mear-Term Investment	N/A Government	Government Required	Undetermined Government Interest	No Government nor Contractor Interes
	Undetermined*	Conntrac:or interent	Conntractor interent	

Long-Term Investment Recovery	N/A Government	Same As Above	Same As Above	Same As Above
	Undetermined*			
Near-Term Competitive Advantage	N/A	Government Required	Undetermined Government Interest	N/A
	Unknown	Yes, if Government Rights Deferred	Yes, if Government Rights Deferred	
Long-Term Competitive Advangage	N/A	Same As Above	Same As Above	N/A
	Unknown			
Elimination of Competition	N/A	Government Required	Undetermined Government Interest	N/A
		Conntractor interent	Conntractor interent	

Table C-8 Anticipated Reuse Environment Matrix

Frequency of Application	**Anticipated Reuse Environments**				
	Development	Test	Maintenance and Support	Modification	Derivatives
Single Program Application	No	No	No	No	No
Multiple, Specific Program Applications	XX	XX	XX	XX	Unknown
Unlimited, Government Program Applications	Undetermined	Undetermined	Undetermined	Undetermined	Undetermined
One-Time Use	No	No	No	No	No

Multiple, but Limited, Use	No	No	No	No	No
Unlimited Use	No	No	No	No	No

The Program Manager and the staff concluded that the strategy of not requiring delivery of unlimited rights until the second year of O&M was still valid. The contracting officer noted, however, that at least the minimum rights required by DFARS 252.227-7013 must be provided. All agreed and noted this would not conflict with the chosen strategy. They also agreed that the RFP would encourage innovative approaches to rights which does not prohibit/impede the following reuse objectives:

- The RFP would reflect that multiple program reuse is anticipated. The PEO directed all these programs (Programs in Table C-2) be identified so industry could fully understand the Government's current thinking.

- There is potential for future, broader Government use of developed components.

Legal counsel also reminded the staff to assure that all documentation associated with the software be acquired so it is consistent with the approach to software rights.

Any licensing and associated royalties should be evaluated consistent with the templates and tables in Section 2.5.4, License Agreement and Section 2.5.5, Royalties of this handbook.

Incentives.A separate award fee, equal to up to 5% of total fee, will be used for software reuse. The fairly large ward fee, for reuse only, serves to emphasize its importance in both the PEO and PM's approaches (See Section 2.5.6, Incentives). The award fee will focus on:

- Adequacy of integration of reuse into the offeror's system and software engineering methodology

- Identification and successful use of credible technical, schedule and cost metrics for reuse

- Delivery of satisfactory reusable components

50% of the award fee will be tied to actual delivery of satisfactory reusable components (see section 6.5 for award fee plan examples).

Evaluation Criteria.Section 6.1 contains examples of criteria to be used. This case study would consider use of all the criteria in examples 1 (6.1.1) and 2 (6.1.2) and their related proposal instructions in 6.2.

Evaluation Standards.Section 6.3 provides samples of evaluation standards. Sample Standard No.1 could be used to evaluate existing components proposed for reuse. Sample Standard No. 2 could be used for development of reusable components. Sample Standard No. 3 could be used to evaluate adequacy of software and data rights proposed.

Note: A standard should be prepared for each evaluation element identification.

The Program Manager proceeded towards a planned reuse approach and contract award.

APPENDIX D - SOFTWARE REUSE SUCCESS STORIES

D.1 Army Command and Control System (ACCS) Common Software Program

Programs within the Army's Tactical Command and Control domain were examined to determine if there were any commonalities for hardware, software, and standards and procedures to which software reuse practices could be applied. Five systems, all in different phases of development, were determined to have commonalities of various functions. In utilizing the common software among these systems, the Army has realized a cost avoidance of $479.9M ($202.4M for common support software and $277.5M for common applications software). As a result, they were able to deal with funding shortfalls while responding to additional mission requirements. These five systems are: All Source Analysis System (ASAS), Combat Service Support Control System (CSSCS), Forward Area Air Defense Command and Control (FAADC2), Maneuver Control System (MCS), and Advanced Field Artillery Tactical Data System (AFATDS).

D.2 Bofors

Bofors, now NobelTech Systems AB, Sweden, utilized sound software engineering principles, the Rational software engineering environment and Ada in designing an embedded shipboard Command, Control, and Communications (C3) application (FS2000). These major modifications in development approach resulted in significantly reduced development costs, improved return on investment, and enhanced overall competitiveness.

Prior to the development of FS2000, as the complexity of development efforts increased, in concert with heightened demands on fewer skilled software engineers, management determined that the software technology currently being utilized was not keeping pace with demands. Consequently, Bofors' management made a conscious decision to modify its long-term strategy by designing a system family rather than numerous specific systems to ensure future competitiveness. Two key ingredients of this new strategy included radical modification of the development process due to the pressing need for increased productivity, and an emphasis on development of common software subsystems for systematic future reuse. Since the C3 application would result in extensive reuse across many differing platforms used by multiple countries, flexibility in software architecture and design was built into the system, which in turn was reflected in developed code. Additionally, the dual goals of optimizing modularity and adaptability in fielded software were implemented. Development risk was also reduced through rapid prototyping of system designs and incremental integration of subsystems. As a result of this novel development approach, productivity doubled (as compared with previous programs), subsequent implementations of FS2000 technology are realizing up to 70% reuse, integration resource allocation has been halved, cost savings are significant ($20 million on first program alone), and the return on their investment in the Rational environment was 470% for FS2000. Ultimately, competitiveness was also enhanced, as Bofors has not lost a major Ada competition or follow-on contract since the development of FS2000 [18].

D.3 Command Center Processing and Display System Replacement (CCPDS-R)

The original version of Network Architecture Services (NAS) was conceived as a TRW Independent Research and Development project, with a requirement for reusability as a design constraint. System development was quite complex, due to functional sophistication, performance criticality, reliability requirements and design-for-reuse. The NAS software was subsequently refined and evolved to facilitate its transformation into a reusable, production quality product, and was applied during development of CCPDS-R. The support of effective Ada environments, top-notch personnel (with no critical turnover), management commitment to the project, and widespread usage of the resulting product by diverse users under different execution environments helped ensure its success.

Network Architecture Services' success in enhancing productivity within CCPDS-R resulted in part from the organization of difficult portions of system development into reliable and powerful generic building blocks that are easy to use. This in turn facilitated development with minimal risk, simple integration, and ease in incorporating modifications resulting from lessons learned during incremental development. Two significant advantages result from the use of NAS: "1) Value added operational software through reuse of mission independent, performance tunable components which support open architectures, and 2) Overall project productivity enhancement as a result of NAS support for rapid prototyping, runtime instruction tool suite, and encapsulation of the difficult capabilities required in any distributed real-time system into a standard set of building blocks with simple applications interfaces." Such benefits result in significant reduction in "applications software complexity, less reliance on scarce real-time programming experts, and a substantial reduction in overall project risk" [19].

D.4 Fujitsu

Fujitsu's Information Support Center library was developed as a result of analyzing their existing electronic switching systems. It is staffed with domain experts, software engineers and reuse experts. Library staff members are included in all design reviews and use of the library is compulsory. Fujitsu has experienced a 20% improvement in schedule from all projects and 70% schedule improvement in electronic switching development [20].

D.5 Frontier Technologies

Frontier developed a real-time remotely piloted vehicle control workstation exclusively in Ada, incorporating 13% reuse through use of Generic Reusable Ada Components for Engineering (GRACE) components for the creation of low-level system components. Frontier's Vice-President explained that "the system was designed to maximize the commonality of functions where possible. This design allowed for the selection of common reusable software components for the software implementation. The GRACE components were selected because the functionality of the components could provide significant cost savings during the implementation phase." Once the software was delivered to the prime, further evidence of its reusability and maintainability was established by the facilitation of comprehension and subsequent update [21].

D.6 GTE Data Services

GTE Data Services has established a corporate-wide reuse program. Its activities include identification of reusable components, development of new components, cataloguing of all components in an automated library, components maintenance, reuse support, and a management support group. GTE reports first year reuse of 14% and savings of $1.5 million, and projects 50% reuse and $10 million savings in telephony management software development [20].

D.7 Hewlett-Packard (Logic Systems Division)

Software reuse has been actively practiced at Logic Systems Division since 1985. Reuse has taken many forms, including emulator building, a configuration management system, and the use of 75 common libraries. The common libraries were initiated by engineering staff, and are owned and managed by specific individuals. Routines are also individually owned, and submissions to specific libraries must conform to specific requirements (e.g., formal write-ups, test procedures). Hewlett-Packard has classified reuse into three categories, depending upon:

1. type of reuse (module, subsystem, large subsystem),

2. level of reuse (individual, within project, between projects, within company, industry), and

3. leveraged (unsystematic) versus pure reuse (systematic code reuse).

The major reuse success within Logic Systems Division occurred as a result of introducing software reuse across an entity (emulators). Generic emulation software was developed for use across all emulators within a family of systems. There is some large subsystem reuse that has taken place, but the vast majority of reuse within this division involves use "as is" of existing products. Although reuse positively impacted productivity and quality in the original development effort, greater effects were realized in subsequent product development, where reuse was leveraged, and entire subsystems might have been lifted and reused. The average amount of reuse which occurred across ten emulator developments approached 74%. Results of their emulator experience led to the conclusion that an average emulator developed entirely anew requires 60% more time to complete than one with an average amount of reuse.

Code reuse was also instituted within their cross compiler product development, which resulted in much lower defect levels, and much shorter development time to produce a "new" compiler. Although there is internal awareness of differing constraints which impact types and levels of reuse (e.g., when moving from individual to industry reuse, the process must become increasingly formalized), formal reuse goals, a formal database of reusable components and a fancy browser have not been established. Despite these deficiencies, their reuse efforts have succeeded, due in part to their configuration management system, good domain analyses of projects, and a positive management attitude toward reuse [22].

D.8 Magnavox

Magnavox modified and reused a significant amount of code during their development of a force fusion system prototype (FFSP) under the Army Advanced Field Artillery Tactical Data System (AFATDS) program. FFSP incorporated automated functions for a command and control system in support of evolving requirements for National Command and Control System Afloat. Ultimately, the goal was to determine the feasibility of reusing software between services. The project required about 20% of the projected estimated time to complete FFSP with totally new development, and Magnavox realized 93.45% reuse of Ada code from AFATDS [21].

D.9 NEC Software Engineering Laboratory

In analyzing its business applications, NEC identified 32 logic templates and 130 common algorithms. As a result, they established a reuse library to catalogue and facilitate use of these templates and components. The library has been automated and integrated into NEC's software development environment, which enforces reuse in all stages of development. NEC reports a 6.7:1 productivity improvement and a 2.8:1 quality improvement [20].

D.10 Raytheon Missile Systems

Raytheon instituted a reuse program after recognizing redundancy in its business application systems. In analyzing over 5000 production COBOL programs, three major classes were identified. Templates with standard architectures were designed for each class, and a library of parts were developed by modifying existing modules to fit the architectures. Raytheon reports an average of 60% reuse and 50% net productivity improvements [20].

D.11 REBOOT

REBOOT, Reuse Based on Object Oriented Techniques, is a consortium of European firms from France, Spain, Germany, Norway, Sweden and Italy, whose objective is to "develop, test and disseminate an industrial environment with associated methods to support software reuse by means of object-oriented technology". Current plans include provision of:

1. a software engineering environment to create, store and retrieve reusable components,
2. a methodology for populating bases with reusable components,
3. a methodology for reusing components,

4. a base of general purpose software components,

5. bases of domain-specific reusable software components, and

6. a study of managerial, organizational and legal aspects.

During a presentation at the STARS/Swedish Software Reuse Information Exchange briefing, plan participants stated that they have currently developed a data model and conceptual schema, metrics attributes for reuse, a global architecture for two segments, a general architecture for the first prototype, and preliminary specifications of reuse tools. Their strategy for populating a specific domain with reusable components encompasses the following: identifying potentially reusable components by domain analysis, cross-product analysis, and reverse engineering of an existing application; extracting essentials of these identified components (in terms of objects); classifying and specifying these components; and implementing by need. Ultimately, REBOOT anticipates providing "the basis for an industry-wide dissemination platform for standardization purposes". REBOOT is also intimately connected with and partially dependent upon many other European reuse projects, such as ESPRIT I and II projects, RACE, EUREKA and National Research Projects underway in Spain, Sweden, Norway, and Italy [11].

D.12 Restructured Naval Tactical Data Systems (RNTDS)

RNTDS was initiated at the Fleet Combat Direction Systems Support Activity (with support from Unisys Systems Corporation) in 1976 to reduce costs associated with "development and maintenance of multiple implementations of functionally similar software in a structured environment". It was also expected to reduce the length of time required to develop and/or modify software and to improve overall quality.

The RNTDS system was designed to define a large number of tasks which could be applied as building blocks for constructing customized programs. A single performance specification was developed for the RNTDS domain, and an initial program baseline was then derived for the lead class of ships. Requirements for other classes which differed from the baseline were defined as deltas to that baseline, rather than as new sets of complete requirements. A reusable parts library was developed to maintain all tasks and data.

The RNTDS methodology has ensured a high degree of reliability, since the same code is currently being utilized across many different ship configurations, which has facilitated rapid error identification. When the Navy introduced a major upgrade to shipboard tactical systems, the Advanced Combat Direction System (ACDS), RNTDS was selected as the architecture to be used in implementing ACDS requirements.

The use of RNTDS has resulted in 26% fewer labor hours than other methodologies in use on equivalently sized programs. Although the original estimate for reuse potential across RNTDS was 60%, actual average results are hovering around 90%, with 77% commonality across programs. The Navy credits the productivity and quality gains to the cooperative relationship between FCDSSA and Unisys, the philosophy of continuous process improvement, the well-defined development methodology, and the willingness of all parties to apply lessons learned [23].

D.13 SofTech, Inc.

In building its Ada compiler products, SofTech, Inc. implements software reuse at the generic architecture level. They develop compilers for new host and target systems by replacing only selected modules from the standard architecture. This high-level reuse has led to a productivity level of 50K lines of code per person-year (10-20 times the industry average) [20].

D.14 Universal Defence Systems

Universal Defence Systems develops Ada command and control applications. The company began its work in this business with a reuse focus, and has developed a company-owned library of 396 Ada modules comprising 400-500 thousand lines of code. With this base, they have developed the Australian Maritime Intelligent Support Terminal with approximately 60% reuse, delivering a 700 thousand lines of code system in eighteen months [20].

D.15 TRILLIUM: Telecom Software Product Development Capability Assessment Model

Bell Canada sells telecommunication (voice, data and image) services to its end customers. Its two primary suppliers are Northern Telecom (NT) and Bell Northern Research (BNR). Bell Canada has utilized software reuse and even had a software reuse library in place for many years, when it determined its processes needed improvement. As a result, a tri-corporate effort was established to assess and enhance NT's and BNR's software development process. Under this Tri-Corporate Team, a software capability improvement model and methodology that includes software reuse as an evaluation factor was developed. This model is TRILLIUM, Telecom Software Product Development Capability Assessment Model. The Tri-Corporate Team also maintains TRILLIUM. Bell Canada is responsible for maintaining the generic architecture and the 3-piece library (Product Library System, Product Development Environment, and the Product Tracking System). NT and BNR are responsible for using TRILLIUM and to develop and maintain the reusable components. As a result of the joint effort, the 3 companies have established a high level of trust in their relationships and each have recovered their investment costs. Each of them also has increased sales and market share, and have gained a competitive advantage from effective software reuse.

APPENDIX E - COMPONENT RECOVERY THROUGH RE-ENGINEERING

Re-Engineering assists in conducting domain analysis, identifying reuse components, and salvaging existing products. This option may be the first step in implementing reuse and may appear less formidable than attacking the reuse initiative with no components. The procedure provides an incremental approach to reuse by allowing leverage of existing products. If components could be salvaged, there may also be some up-front savings.

The component recovery effort should include the following steps:

Inventory existing systems.

Look for "similar" code.

Determine if economically feasible to recover components.

Apply similarity criteria.

Encapsulate similar code/data as reusable parts.

Certify these parts.

Put Parts in repository.

Monitor reusable part usage.

Refine process.

APPENDIX F - GLOSSARY

Classification Scheme	The organization of reusable software components within a reuse library system according to specific criteria.
Cooperative Research and Development Agreement	A type of licensing arrangement in which government helps transition its technology, but does not provide financial assistance.
Command Center	A facility from which a commander and his representatives direct operations and The process of confirming that a component correctly implements its stated function(s), adheres to quality and reuse standards. Measures a component's goodness of fit into the applicable domain control forces. It is organized to gather, process, analyze, display and disseminate planning and operational data and to perform other related tasks [24].
Component	A software resource, which is usually developed during a software development. These include: requirement, designs, specifications, code, test cases, and documentation.
Certification	The process of confirming that a component correctly implements its stated function(s), adheres to quality and reuse standards. Measures a component's goodness of fit into the applicable domain.
Copyright	The legal right to reproduce, publish and sell a product.
Domain	The set of current and future systems/subsystems marked by a set of common capabilities, data and application. The application area made up these systems/subsystems.
Domain Analysis	The process of identifying, collecting, organizing, analyzing, and representing a domain model and software architecture from the study of existing systems, underlying theory, emerging technology, and development histories within the domain of interest. A commonality study which identifies the similarities and differences among related systems in an application area.
Domain Engineering	An entire life cycle engineering process conducted in a domain (application area) that includes domain analysis and the subsequent construction of components, methods, tools, and supporting documentation.
Domain Expert	An individual who is knowledgeable in a particular domain. These people can be experienced users or

	software developers of systems in the domain. The domain expert may or may not be familiar with domain engineering.
Domain Knowledge	The collection of functions, components, data, and requirements for a domain.
Domain Management	The function of centrally managing software reuse for a particular domain (either horizontal or vertical). This role includes: developing software reuse strategies, conducting requirements and domain analyses, and maintaining the domain knowledge and associated components for the domain.
Domain Model	A formal, concise representation of a domain's functions, components, data, requirements, relationships and variations. Identifies the generic requirements, represents the formal definition of the domain, and provides the general rules and principles for operating within the domain. It indicates the boundaries of the domain, the primary inputs and outputs and the standard vocabulary used.
Domain-Specific Library	A library which is driven and built around a domain that has been carefully scoped and modeled.
Domain-Specific Software Reuse	Reusing components in a specific domain (through the use of a domain-specific library) to build an instance of an application in that domain.
Generic Architecture	The high-level design for a software system or subsystem, that includes the description of each software components' functionality (or result), name, parameters and their types and a description of the components' interrelationships for widespread use within a particular domain. The structure and relationship among the components of a system that is sufficiently generalized to enable application throughout a domain.
Generic Design	Development of a design for a family of systems, rather than for one specific application
High-level Components	Components that are general to the entire domain, such as domain model or generic architecture
Horizontal Domain	A domain that consists of a particular kind of software process (e.g., user interface) that has applicability across

	vertical domains. An application-independent grouping of software (e.g., data structures, general math routines).
Liability	A legal responsibility to correct defects of a product.
License Agreements	An agreement between an owner and user of a product which identifies the rights and obligations of each party for use of the product.
Low-level Components	Components that are specific to a particular system, although they can still be reusable in other systems. These are: requirements, specifications, test plans, procedures and results, object code, source code, or system/software documentation.
Non-Developmental Item	Any item available in the commercial marketplace (i.e., COTS); Any previously developed item in use by Government (i.e., GOTS); Any COTS or GOTS that requires only minor modification to meet the requirements of the procuring agency; Any item currently being produced that does not meet the above because it is not yet in use or is not yet available in the commercial marketplace (DoD-D-5000.1).
Patents	An exclusive legal ownership of a product, which is granted by the U.S. Patent Office.
Patent Disclosure	Notice of the existence of a patent. Existence of patents are publicly disclosed only after the patent is approved, not while during the approval process.
Product Design	Derived from the specification of the architecture, describes the relationship between the domain model and the implementation components.
Recoupment	A contract clause which requires a contractor to payback to the Government development dollars from a previous contract when a product from that contract is subsequently sold commercially (non- Government sale).
Royalties	Payments to a party for use of its property: paid-up royalty, a lump- sum amount, paid at one time; running royalty, a set amount paid over the term of the license; set period royalty, a set amount paid over a pre-agreed period of time.
Software Engineering Environment	The computer hardware, operating system, tools, computer-hosted capabilities, rules and techniques that

	support development of software throughout the life cycle.
Software Reuse	A process in which software resources are applied to more than one system.
Warranty	An obligation to repair/replace product found to be defective during the period of warranty.
Vertical Domain	A domain that addresses all levels of a single application area across functional lines (e.g., information systems, command and control, weapon systems, command centers, or battle field movement). Vertical domains are sometimes drawn along program management (i.e., PEO/ DAC) boundaries.

APPENDIX G - ACRONYMS

ACCS	Army Command and Control System
ACWP	Actual Cost of Work Performed
AFATDS	Advanced Field Artillery Tactical Data System
AFDO	Award Fee Determining Official
AFRB	Air Force Review Board
ARC	Army Reuse Center
ASAS	All Source Analysis System
ASPM	Armed Services Pricing Manual
ASSET	Asset Source for Software Engineering Technology
ATCCS	Army Tactical Command and Control System
BCWP	Budgeted Cost of Work Performed
CAMP	Common Ada Missile Packages
CASS	Common ATCCS Support Software
CARDS	Central Archive for Reusable Defense Software
CDRL	Contract Data Requirements List
CECOM	Communications - Electronics Command (U.S. Army)
CHS	Common Hardware and Software
COTS	Commercial-Off-The-Shelf
CPR	Cost Performance Reports
CPIF	Cost Plus Incentive Fee
CRDA	Cooperative Research and Development Agreement
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DAC	Designated Acquisition Commander
DFARS	DoD Federal Acquisition Regulation Supplement

DISA	Defense Information Systems Agency
DSRS	Defense Software Repository System
ESC	Electronic Systems Center (U.S. Air Force)
FAR	Federal Acquisition Regulation
GFE	Government Furnished Equipment
GOTS	Government-Off-The-Shelf
GPALS	Global Protection Against Limited Strikes
GRACE	Generic Reusable Ada Components for Engineering
IFPP	Instructions for Proposal Preparation
IOC	Initial Operating Capability
JAST	Joint Advanced Strike Technology
JIAWG	Joint Integrated Avionics Working Group
JLC	Joint Logistics Commanders
MIRRR	Most Important Reuse Requirements or Risks
NDI	Non-Developmental Item
NRC	Non-Recurring Costs
O&S	Operations and Support
OSD	Office of the Secretary of Defense
PCO	Procuring Contracting Officer
PDL	Program Design Language
PDR	Preliminary Design Review
PDSS	Post-Deployment Software Support
PEO	Program Executive Officer
PM	Program Manager
PRISM	Portable Reusable Integrated Software Modules
RAASP	Reusable Ada Avionics Software Packages
RAAT	Reuse Acquisition Action Team

RAPID	Reusable Ada Products for Information Systems Development
RFP	Request For Proposal
SBIR	Small Business Innovative Research
SDIO	Strategic Defense Initiative Organization
SDP	Software Development Plan
SEE	Software Engineering Environment
SIGAda	Special Interest Group - Ada
SLOC	Source Lines Of Code
SRP	Software Reuse Plan
STSC	Software Technology Support Center
USAISEC	U.S. Army Information Systems Engineering Command

APPENDIX H - REFERENCES

[1] DoD Software Reuse Vision and Strategy, Document #1222-04- 210/40, 15 Jul 9

[2] Waste Not, Want Not: Recycling Software Promises a Different Kind of "Green" Solution, Paul Kinnucan, Paramax Performance, 92

[3] Criteria for Comparing Domain Analysis Approaches, draft, Steven Wartik, Ruben Prieto-Diaz, Software Productivity Consortium, 1991

[4] Current FAR and Budget/Finance Environments, STARS-SC- 03501/001/00, 30 Mar 91

[5] Reusable Software Components, Dr. Trudy Levine, Fairleigh Dickinson University, STSC CrossTalk, Mar 1992

[6] Software Effort and Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information, CMU- SEI-92-TR-21, ESC-TR-92-021, Sep 92

[7] Software Quality Measurement: A Framework for Counting Problems and ·fects, CMU-SEI-92-TR-22, ESC-TR-92-022, Sep 92

[8] Software Size Measurement: A Framework for Counting Source Statement, CMU-SEI- 92-TR-20, ESC-TR-92-20, Sep 92

[9] Software Measurement Concepts for Acquisition Program Managers, CMU-SEI-92-TR- 11, ESC-TR-92-11, Jun 92

[10] A Concept Study for a National Software Engineering Database, CMU-SEI-92-TR-23, ESC-TR-92-023, July 92

[11] Risk-Reduction Reasoning-Based Development Paradigm Tailored to Navy C2 Systems, TRW, STARS-SC-03070/001/00, 30 Jul 9

[12] Statement of Work for the Portable, Reusable, Integrated Software Modules (PRISM) Program, 15 Jul 91, ESD/AVS, Hanscom AFB, MA

[13] Impact of Domain Analysis on Reuse Methods - Final Report CECOM, Center for Software Engineering, Advanced Software Technology, CIN:C04-0872D-0001-00, 6 Nov 89

[14] Software Management Metrics, Herman P. Schultz, MITRE Corp., ESD-TR-88-001, May 88

[15] Joint Integrated Avionics Working Group (JIAWG) Contract Elements for Software Reuse (Draft), Prepared by (JIAWG)Software Task Group, 13 Jun 90

[16] The Army Strategic Software Reuse Plan, Final, 31 Aug 92

[17] DoD Systems Acquisition and Software Reuse Working Group Meeting Minutes, 5-7 Nov 91, held at the Institute for Defense Analysis, Arlington, VA

[18] Foundation for Competitiveness and Profitability: FS2000 System , Rational, and Ada; Rational, Aug 91

[19] Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-Task Networks: Network Architecture Services (NAS), TRW Systems Engineering and Development Division, Dec 89

[20] NATO Standard for Management of a Reusable Software Component Library, Volume 2, NATO Communications and Information Systems Agency

[21] Software Reusability - A DACS State-of-the-Art Report, Valentine W. Tirman, Jr., Productive Data Systems, 1 Aug 90

[22] The Successful Introduction of Software Reuse Across an Entity, Jim Jost, from the 16th Annual Software Engineering Workshop, NASA Goddard Software Engineering Laboratory, 3-4 Dec 91

[23] Deriving and Managing Functional Commonality for Software Reuse: An RNTDS Experience, Jim Aridas, Paramax Systems Corporation, CrossTalk, Feb 92

[24] DoD Dictionary of Military and Associated Terms, Joint Publication, 1-02, 1 Dec 87

[25] Air Force Reuse Strategy (Draft), 1 Apr 92

[26] All Source Analysis System Request For Proposal, Army Tactical Command and Control System

[27] A Program Manager's Guide to Generic Architecture's, Final Report, CECOM, Center for Software Engineering, Advanced Software Technology, CIN C04-N-0001-00, 10 May 89

[28] Bush Ends Arms Export R&D Recoupment Fee, Defense News, Volume 7, No. 25, 22-28 Jun 92

[29] DoD-STD-2167A, Defense System Software Development, 29 Feb 88

[30] DoD Directive 5000.1, Defense Acquisition, 23 Feb 91

[31] Global Protection Against Limited Strikes (GPALS) Software Reuse Strategy, DoD Strategic Defense Initiative Organization, Feb 92

[32] Global Protection Against Limited Strikes (GPALS) Contract Requirements Packages (CRPs) Guidelines for Computer Resource Issues, SDI-S-SD-92-000005, 30 Mar 92

[33] Reuse Metrics and Measurement Concept, Draft, Joint Integrated Avionics Working Group (JIAWG) Prepared by (JIAWG) Software Task Group, 28 Sep 90

[34] A Software Reuse Maturity Model, STSC Conference, Apr 92, Phil Koltun, Anita Hudson, Harris Corporation

[35] Reusability Process Report for the Portable, Reusable, Integrated Software Modules (PRISM) Reusability Process Report, Draft, 27 May 92Reusability Process Report for the Portable, Reusable, Integrated Software Modules (PRISM) Reusability Process Report, Draft, 27 May 92

[36] Software Reuse Handbook, (Annotated Outline), Reifer Consultants, Inc., Joint Integrated Avionics Working Group (JIAWG) Reusable Software Program, 27 Sep 91

[37] Strategic Defense System (SDS) Computer Resources Guidelines for Contract Requirements Packages (CRPs), SDIO, 11 Dec 91

[38] Domain Analysis Bibliography, Software Engineeri [illegible] Institute, CMU/SEI-90-SR-3, Jun 90

[39] Design for Reuse Handbook, SIGAda Reuse Working Group

[40] Domain Analysis Process, Interim Report - Domain Analysis Project, Software Productivity Consortium, DOMAIN_ANALYSIS-90001-N, Version 01.00.03, Jan 90

[41] STARS Reusability Guidelines, Prepared by IBM for Electronic Systems Division, 30 Apr 90

[42] Reuse Library Process Model, Software Technology for Adaptable, Reliable Systems (STARS) Program, IBM Federal Sector Division, 26 Jul 91

[43] Software Reuse Guidelines, U.S. Army Information Systems Engineering Command (USAISEC), U.S. Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), Apr 90

[44] Force Software Release Policy, USAF DCS/C4, 23 Jun 93

[45] Model Contracts/Agreements, Central Archive for Reusable Defense Software (CARDS), STARS-VC-B014/001/00, 25 Mar 94

[46] DoD Federal Acquisition Regulation Supplement: DFARS

[47] Software Reuse Project Officer (RPO) - Information Memorandum SAF/AQKS, 26 Feb 93

[48] Software Reuse Incentive Policy, SAF/AQ Policy Memo 93M007 4 Jun 93